ÉTR · TRAITÉ DE RÉFÉRENCE

Ligne cognitive d’une IA

ÉTR — Équation des transformations relationnelles

Édition 2026

ÉTR · Représentation → Comparaison → Transformation → Décision → Mémoire

PREMIÈRE PARTIE

Pourquoi un nouveau langage relationnel ?

Chapitre 1 Comparer est plus difficile qu'il n'y paraît

« Toute intelligence compare. Pourtant, peu de systèmes décrivent réellement leurs comparaison. » Lorsqu'un être humain observe deux objets, il a rarement l'impression d'effectuer un calcul. Il reconnaît immédiatement des ressemblances, des différences, des analogies, des ruptures ou des continuités. Deux visages semblent appartenir à une même famille. Deux musiques évoquent une émotion similaire. Deux textes parlent du même sujet, bien qu'ils n'utilisent presque aucun mot identique. Deux entreprises poursuivent des stratégies comparables malgré des secteurs d'activité très différents. Cette capacité paraît naturelle. Pourtant, elle constitue l'un des problèmes les plus complexes de l'informatique moderne. Car un ordinateur ne perçoit ni ressemblances, ni intentions, ni significations. Il manipule uniquement des représentations. Toute comparaison commence donc par une question plus fondamentale : Comment représenter une information de manière à pouvoir ensuite la comparer ? Cette question précède toutes les autres. Avant de reconnaître une image, il faut savoir comment la représenter. Avant de comparer deux phrases, il faut décider ce qu'est une phrase pour la machine. Avant de retrouver un document, il faut choisir la manière dont ce document sera décrit. Avant de produire une réponse, il faut construire une représentation interne du contexte. Autrement dit, toute intelligence artificielle repose sur deux problèmes distincts : . Comment représenter le monde ?

. COMMENT COMPARER CES REPRÉSENTATIONS ?

Ces deux questions sont souvent étudiées ensemble, alors qu'elles correspondent à deux difficultés différentes.

La première concerne la description. La seconde concerne la relation.

Une même réalité, plusieurs représentations Imaginons une situation très simple. Une personne prononce la phrase : Le chat dort sur le canapé. Pour un être humain, cette phrase semble unique. Pour un ordinateur, elle peut pourtant être représentée de nombreuses manières. Comme une suite de caractères : L e

c h a t ... Comme une suite de codes Unicode. Comme une suite de bits. Comme une liste de mots. Comme un arbre syntaxique. Comme un vecteur. Comme un ensemble de probabilités. Comme un graphe. Comme une séquence de transformations. Aucune de ces représentations n'est fausse. Elles répondent simplement à des objectifs différents. C'est probablement l'idée la plus importante à retenir avant d'aborder l'intelligence artificielle moderne. Une représentation n'est jamais la réalité. Elle est une manière de rendre cette réalité exploitable pour résoudre un problème particulier.

Pourquoi existe-t-il autant de modèles différents ? Cette diversité surprend souvent. Pourquoi les chercheurs ont-ils construit des CNN, puis des Transformers, puis des GNN, puis des moteurs

vectoriels, puis des graphes de connaissances ? Pourquoi ne pas avoir développé un unique modèle capable de tout faire ? La réponse est simple. Parce que les problèmes ne sont pas les mêmes.

Reconnaître un visage n'est pas retrouver un document. Retrouver un document n'est pas dialoguer avec un utilisateur. Dialoguer n'est pas piloter un robot. Piloter un robot n'est pas organiser la mémoire d'une entreprise. Chaque architecture est donc une réponse spécialisée à une famille de problèmes. Le succès actuel des grands modèles de langage ne doit pas masquer cette réalité. Les Transformers excellent dans certaines tâches. Les réseaux convolutifs restent parmi les meilleurs outils pour de nombreuses applications de vision. Les graphes de connaissances permettent de représenter explicitement des relations que les modèles statistiques ne décrivent pas directement. Les moteurs de recherche documentaires demeurent extrêmement performants lorsqu'il s'agit d'indexer plusieurs milliards de documents. Aucun de ces systèmes n'est universel. Ils sont complémentaires. Comprendre cette complémentarité est indispensable avant d'introduire l'ÉTR.

Comparer n'est pas seulement mesurer Lorsque nous comparons deux objets, nous avons souvent tendance à imaginer une distance. Deux villes peuvent être séparées de cent kilomètres. Deux couleurs peuvent être proches dans un espace colorimétrique. Deux vecteurs peuvent former un angle faible. Deux nombres peuvent différer de quelques unités. Ces comparaisons sont utiles. Mais elles ne décrivent qu'une partie du problème. Supposons maintenant les deux phrases suivantes : Le médecin examine le patient. Le chirurgien ausculte le malade. Une représentation fondée uniquement sur les mots constatera que les deux phrases utilisent un vocabulaire différent. Une représentation vectorielle pourra apprendre qu'elles sont proches. Un graphe pourra relier les concepts. Un modèle de langage pourra produire une interprétation contextuelle. Toutes ces approches apportent une réponse pertinente. Mais elles ne décrivent pas nécessairement quelles transformations permettent de passer d'une représentation à l'autre. Autrement dit, elles répondent principalement à la question : À quel point ces deux objets se ressemblent-ils ? L'ÉTR ajoute une question différente :

Par quelles transformations relationnelles ces deux objets deviennent-ils comparables ? La différence paraît subtile. Elle est pourtant profonde. Comparer une distance revient à observer un résultat. Comparer une transformation revient à observer un processus.

Une autre manière de regarder les systèmes Pendant plusieurs décennies, la plupart des progrès de l'intelligence artificielle ont consisté à améliorer la manière de représenter les données. Des représentations plus riches. Des représentations plus compactes. Des représentations apprises automatiquement. Des représentations contextuelles. L'ÉTR ne remet pas en cause cette évolution. Elle propose simplement de déplacer le centre de gravité. Au lieu de considérer que la représentation constitue l'objet principal du calcul, elle considère que ce sont les transformations entre représentations qui méritent d'être décrites comme des objets scientifiques à part entière. Ce changement de perspective ne remplace pas les approches existantes. Il les complète. Une image peut toujours être représentée par un CNN. Une phrase peut toujours être représentée par un Transformer. Un document peut toujours être indexé par un moteur de recherche. L'ÉTR intervient lorsque l'on souhaite décrire, conserver, comparer et auditer les transformations qui relient ces représentations, indépendamment de la technologie qui les a produites.

Ce que le lecteur découvrira dans ce livre Les chapitres suivants présenteront progressivement les principales familles d'intelligence artificielle utilisées aujourd'hui. Chacune sera étudiée selon la même méthode :

- le problème auquel elle répond ;
- la manière dont elle représente l'information ;
- la façon dont elle réalise les comparaisons ;
- ses points forts ;
- ses limites intrinsèques ;
- les situations dans lesquelles elle est particulièrement adaptée.

Ce n'est qu'après ce parcours que l'Équation des Transformations Relationnelles sera introduite. Ce choix est volontaire. Comprendre une nouvelle théorie est beaucoup plus facile lorsqu'on comprend d'abord les questions auxquelles les théories existantes répondent déjà avec succès. L'objectif de ce livre n'est donc pas d'opposer les approches, mais de montrer qu'elles occupent des places différentes dans un paysage scientifique beaucoup plus vaste, où la représentation, la comparaison et la transformation constituent trois problèmes complémentaires plutôt qu'un seul.

PREMIÈRE PARTIE

Pourquoi un nouveau langage relationnel ?

Chapitre 2 — Représenter n’est pas expliquer

Une intelligence artificielle ne traite jamais directement un objet du monde. Elle traite une représentation calculable de cet objet. Une photographie devient un tableau de valeurs numériques. Une phrase devient une séquence de symboles ou de vecteurs. Une entreprise devient un ensemble de données, de catégories, de flux et d’indicateurs. Une personne devient, selon le système considéré, un profil, un historique d’actions, un ensemble de préférences ou un nœud dans un réseau. Cette conversion est indispensable. Un calcul ne peut commencer que lorsque l’objet a été transformé en une forme compatible avec les opérations du système. Mais une difficulté devient visible immédiatement : La représentation sélectionne déjà ce que le système sera capable de voir. Une représentation n’est donc jamais neutre. Elle conserve certaines propriétés, en simplifie d’autres et en rend certaines impossibles à retrouver sans certaines clefs. Considérons une image réduite à sa couleur moyenne. Cette représentation permet de comparer rapidement la dominante chromatique de plusieurs images. Elle ne permet plus de savoir si l’image contenait un visage, un paysage ou un texte. Considérons maintenant une phrase réduite à la fréquence de ses mots. Cette représentation permet d’identifier certains thèmes. Elle ne conserve pas nécessairement l’ordre des mots, la négation, la syntaxe ni la progression argumentative. Dans les deux cas, l’objet représenté existe encore sous une forme calculable, mais une partie de sa structure a disparu.

La première exigence d’un système scientifique rigoureux devrait donc être la suivante :

é é

Cette exigence paraît simple. Elle est pourtant rarement appliquée de manière complète.

2.1. La représentation comme transformation initiale

Soit un objet (x) appartenant à un domaine (\mathcal{X}).

Une représentation peut être décrite par une application :

où :

- (\mathcal{X}) est le domaine des objets initiaux ;
- (\mathcal{Y}) est le domaine des représentations calculables ;
- (\Phi(x)) est la représentation de (x).

Par exemple :

transforme une image en tenseur numérique. De même :

transforme une phrase en séquence de tokens. Mais cette écriture ne suffit pas. Elle indique seulement qu'une transformation existe. Elle ne nous dit pas quelles propriétés ont été conservées.

Pour rendre la transformation scientifiquement lisible, il faut lui associer un contrat. Nous pouvons écrire :

avec :

- (I_c) : informations conservées ;
- (I_t) : informations transformées ;
- (I_a) : informations ajoutées ;
- (I_s) : informations supprimées ;
- (I_{\varnothing}) : informations devenues indéterminables.

Ce contrat ne décrit pas encore l'ÉTR dans son ensemble. Il introduit seulement l'une de ses disciplines fondamentales : une opération n'est pas entièrement définie tant que ses effets informationnels ne le sont pas.

2.2. La différence entre perte, transformation et indétermination

Ces trois notions sont souvent confondues. Transformation Une information est transformée lorsqu'elle change de forme tout en restant récupérable, au moins partiellement. Par exemple, une image convertie d'un espace colorimétrique RGB vers un espace Lab change de représentation, mais une transformation inverse peut théoriquement être définie avec une précision élevée. Suppression Une information est supprimée lorsqu'elle est volontairement retirée. Par exemple, anonymiser un document en supprimant les noms propres constitue une suppression déclarée. Indétermination Une information devient indéterminable lorsque plusieurs objets différents

produisent la même représentation et qu'il n'est plus possible de savoir lequel était l'objet initial. Soit :

Alors la lecture inverse n'est pas unique.

ne désigne plus un objet précis, mais un ensemble de candidats possibles. Cette distinction est décisive. Une perte déclarée peut être acceptable. Une transformation inversible peut être maîtrisée. Une indétermination non déclarée peut produire une illusion de précision.

2.3. Compression et décision

Les systèmes d'intelligence artificielle compressent continuellement l'information. Un CNN compresse progressivement une image en caractéristiques. Un Transformer transforme une séquence de tokens en représentations contextuelles distribuées. Un moteur vectoriel représente un document par un vecteur de dimension finie. Un système de recommandation condense l'historique d'un utilisateur en variables exploitables. Cette compression n'est pas un défaut. Elle est nécessaire. Sans compression, un système devrait conserver et recalculer tous les détails de toutes les entrées à chaque opération. Une telle architecture deviendrait rapidement inutilisable. Le problème n'est donc pas la compression. Le problème apparaît lorsque la compression et la décision deviennent indistinguables. Considérons un système qui transforme un document (d) en vecteur (v_d), puis compare ce vecteur à une requête (q).

Puis :

Le score (s) est souvent utilisé comme base de classement. Mais il faut distinguer trois opérations : . représenter le document ;

. comparer les représentations ;

. décider du classement.

Formellement :

où :

- (\Phi) est une lecture ou une représentation ;
- (\rho) est une comparaison ;
- (\Delta) est une règle de décision.

Ces trois opérations ne sont pas équivalentes. Un bon vecteur ne garantit pas une bonne métrique. Une bonne métrique ne garantit pas une bonne décision.

Une bonne décision locale ne garantit pas un bon comportement global. L'ÉTR exige donc que ces niveaux restent séparés.

é é

2.4. Exemple : la négation

Prenons deux phrases : Le traitement est efficace. Le traitement n'est pas efficace. Elles partagent presque tous leurs mots. Une représentation naïve fondée sur la fréquence lexicale peut les considérer comme très proches. Cette proximité n'est pas fausse. Les deux phrases parlent bien du même sujet. Mais leur orientation propositionnelle est opposée. Cela signifie qu'une seule relation de similarité ne suffit pas à les décrire. Nous pourrions distinguer :

- proximité lexicale ;
- identité du sujet ;
- opposition logique ;
- compatibilité argumentative ;
- similarité syntaxique.

Ainsi, la relation entre les deux phrases n'est pas correctement résumée par un nombre unique. Elle ressemble davantage à une structure :

avec, par exemple :

é

é

é é é

Une moyenne de ces dimensions pourrait produire un score intermédiaire. Mais ce score effacerait la propriété la plus importante : les phrases portent des conclusions opposées. C'est ici qu'apparaît une limite générale des agrégations prématurées.

é ê é

2.5. Un vecteur n'est pas une explication

Les représentations vectorielles sont extrêmement puissantes. Elles permettent de situer des objets dans des espaces où la proximité devient calculable. Mais un vecteur, à lui seul, ne constitue pas une explication. Supposons qu'un modèle associe à une phrase un vecteur :

Ce vecteur peut encoder de nombreuses propriétés distribuées entre ses composantes. Cependant, les questions suivantes restent distinctes :

- pourquoi cette phrase occupe-t-elle cette zone de l'espace ?
- quelles propriétés ont déterminé sa proximité avec une autre phrase ?
- quelle transformation permet de passer de l'une à l'autre ?
- quelle partie du résultat dépend du contexte ?
- quelles dimensions peuvent être interprétées ?
- quelles informations sont absentes ?

Une représentation peut être prédictive sans être directement explicative. Cette distinction est essentielle pour comprendre les LLM. Les grands modèles de langage ne stockent pas généralement leurs connaissances sous la forme d'une encyclopédie explicite où chaque information serait localisée dans une cellule identifiable. Leur comportement résulte d'une distribution complexe de paramètres et d'activations, de sélection d'argument et score appliqués. Ils peuvent donc produire une réponse correcte sans fournir spontanément la chaîne causale exacte qui a conduit à cette réponse. L'explication produite après coup est elle-même une génération du modèle. Elle ne doit pas être automatiquement confondue avec une trace complète de son calcul interne.

2.6. La question de l'inversion

Une représentation est parfaitement inversible lorsque l'objet initial peut être reconstruit sans ambiguïté.

Dans la pratique, de nombreuses représentations ne sont que partiellement inversibles. Par exemple :

- un texte tokenisé peut souvent être reconstruit avec une grande

fidélité ;

- une phrase réduite à un sac de mots ne conserve pas son ordre exact ;
- une image réduite à un histogramme ne conserve pas la disposition

spatiale ;

- un document converti en embedding ne peut généralement pas être

reconstruit mot pour mot.

Cette propriété influence profondément l'usage du système. Une représentation non inversible peut être excellente pour classer et médiocre pour auditer. Une représentation compressée peut être

excellente pour rechercher et insuffisante pour justifier. Une représentation locale peut être excellente pour reconnaître un motif et inadéquate pour décrire une causalité globale. Il n'existe donc pas de représentation universellement supérieure. Il existe des représentations adaptées à des contrats d'usage.

2.7. Le problème du référentiel implicite

Toute représentation est produite relativement à un système de conventions. Une couleur dépend d'un espace colorimétrique. Une position dépend d'un système de coordonnées. Une mesure dépend d'une unité. Un score dépend d'une fonction. Une proximité dépend d'une métrique. Une interprétation dépend d'un contexte. Pourtant, les systèmes informatiques manipulent souvent ces objets comme si leur sens était intrinsèque. Considérons la valeur :

Que signifie-t-elle ? Sans référentiel, nous ne le savons pas. Il peut s'agir :

- d'une probabilité ;
- d'un score de similarité ;
- d'un taux de confiance ;
- d'une précision ;
- d'un coefficient de corrélation ;
- d'une valeur normalisée ;
- d'un seuil interne.

La valeur ne devient interprétable qu'avec son contexte :

é é

De même, une relation ne devient exploitable que lorsqu'on connaît le référentiel dans lequel elle a été calculée.

L'ÉTR nomme explicitement ce référentiel. Il ne s'agit pas nécessairement d'un espace géométrique. Un référentiel peut contenir :

- un domaine d'objets comparables ;
- des règles de transformation ;
- des unités ;
- des conventions ;
- une mémoire ;
- des frontières ;
- des critères d'admissibilité.

Nous pouvons le représenter provisoirement par :

où :

- (\mathcal{O}) désigne les objets admissibles ;
- (\mathcal{F}) les transformations autorisées ;
- (\mathcal{M}) les méthodes de mesure ;
- (\mathcal{B}) les frontières du domaine ;
- (\Pi) les informations de provenance.

Deux résultats numériques identiques produits dans deux référentiels différents ne sont donc pas nécessairement équivalents.

2.8. Ce que signifie réellement « comprendre »

Dans le langage courant, comprendre signifie souvent être capable de répondre correctement. Mais plusieurs niveaux doivent être distingués. Reconnaissance Le système identifie une forme connue. Association Le système relie une entrée à des éléments proches ou statistiquement compatibles. Prédiction Le système estime la sortie la plus probable selon son état et ses paramètres. Justesse Le système reçoit un signal de réalisation, de succès selon son référentiel et nuances. Explication Le système produit une description intelligible des facteurs ayant conduit au résultat. Traçabilité Le système conserve les transformations effectives ayant produit ce résultat. Réversibilité Le système permet de revenir, au moins partiellement, vers les états antérieurs. Un modèle peut être performant en reconnaissance et faible en traçabilité. Il peut être performant en prédiction et limité en réversibilité. Il peut produire des explications linguistiquement convaincantes sans que celles-ci

soient des journaux exacts de ses opérations internes. L'ÉTR ne propose pas de redéfinir arbitrairement l'intelligence. Elle impose seulement de ne pas confondre ces capacités.

é è

2.9. Première conséquence pour l'ÉTR

L'ÉTR ne doit pas seulement représenter des objets. Elle doit représenter les opérations qui affectent ces objets. Si (x) devient (y), il ne suffit pas de sauver (x) et (y). Il faut également préserver, lorsque cela est applicable :

ainsi que le contrat de (f). On obtient alors :

où :

- (x) est l'état initial ;
- (f) est la transformation ;
- (y) est l'état obtenu ;
- (R) est le référentiel ;
- (\Pi) est la provenance ;
- (U) est l'incertitude.

Cet ensemble forme une unité de lecture plus riche qu'un simple couple entrée-sortie. Il devient possible de demander :

- la transformation était-elle admissible ?
- était-elle inversible ?
- a-t-elle conservé l'ordre ?
- a-t-elle supprimé une information ?
- dépendait-elle du contexte ?
- aurait-elle produit le même résultat dans un autre référentiel ?
- quelle incertitude affecte la lecture ?

Ces questions fondent progressivement le langage de l'ÉTR.

Chapitre 3 — Comparer, transformer, composer

Comparer deux objets est souvent présenté comme une opération élémentaire. On choisit une métrique, on calcule une distance, puis on classe les résultats. Cette méthode fonctionne remarquablement bien dans de nombreux contextes. Mais elle repose sur une hypothèse rarement énoncée : les objets sont déjà comparables. Or la comparabilité n'est pas toujours donnée. Elle doit parfois être construite. Comparer deux températures exprimées dans des unités différentes exige une conversion. Comparer deux documents rédigés dans des langues différentes exige une traduction, une représentation commune ou une médiation. Comparer une image et une phrase exige une architecture multimodale capable de les projeter dans un espace partagé. Comparer deux stratégies d'entreprise exige de définir des dimensions, des horizons temporels, les règles, codes et les critères communs. Ainsi, avant la mesure, il existe souvent une transformation.

puis seulement :

La comparaison dépend donc des transformations qui ont rendu les objets comparables.

3.1. Le prédicat de comparabilité

Soient deux objets (x) et (y). Nous définissons un prédicat :

Il indique si (x) et (y) peuvent être comparés dans le référentiel (R). La comparabilité peut dépendre de plusieurs conditions :

- type compatible ;
- unité compatible ;
- niveau de granularité compatible ;
- temporalité compatible ;
- provenance suffisante ;
- représentation disponible ;
- transformation de médiation admissible.

On peut donc écrire :

où chaque (c_k) représente une condition. Dans les systèmes réels, cette valeur n'est pas toujours binaire. Deux objets peuvent être comparables lexicalement et incomparables causalement. Ils peuvent être comparables à court terme et incomparables à long terme. Ils peuvent être comparables selon une dimension et indéterminés selon une autre. La comparabilité doit donc parfois être structurée :

3.2. Relation et transformation

Une relation décrit une correspondance, un rapport ou une propriété entre des objets. Une transformation décrit une opération produisant un changement. Ces notions sont liées, mais elles ne sont pas identiques. Une relation peut être descriptive :

Une transformation est opératoire :

Dans certains cas, la relation peut définir une transformation. Dans d'autres, elle ne fait que caractériser les objets. Par exemple :

—

décrit le rapport entre deux valeurs positives. Mais savoir que :

—

ne précise pas nécessairement le mécanisme physique ou causal ayant transformé (a) en (b). Le passage de (a) à (b) peut résulter :

- d'un doublement ;
- de deux opérations successives ;
- d'une substitution ;
- d'une mesure dans une autre unité ;
- d'un changement de référentiel.

La même relation finale peut donc être produite par plusieurs transformations.

avec :

Cette distinction est fondamentale pour la traçabilité.

3.3. La transformation comme objet déclaré

Dans l'ÉTR, une transformation ne doit pas être réduite à sa fonction d'application. Nous pouvons la représenter comme un objet :

où :

- (\operatorname{type}) indique la famille de transformation ;
- (\operatorname{dom}) est le domaine ;
- (\operatorname{cod}) est le codomaine ;
- (\operatorname{apply}) est la règle d'application ;
- (\operatorname{pre}) contient les préconditions ;
- (\operatorname{post}) contient les postconditions ;
- (\operatorname{inv}) décrit l'inversibilité ;
- (\Pi) conserve la provenance ;
- (U) décrit l'incertitude.

Cette structure rend explicites des propriétés souvent enfouies dans le code ou supposées par le lecteur.

3.4. Les familles de transformations

Plusieurs familles doivent être distinguées. Lecture Une lecture extrait ou construit une représentation.

Transport Un transport déplace un état dans un espace ou un référentiel sans nécessairement changer sa nature.

Qualification Une qualification ajoute ou modifie une lecture sémantique, contextuelle ou normative.

Substitution Une substitution remplace un élément par un autre selon une règle déclarée.

Mise à jour Une mise à jour modifie une mémoire ou un référentiel.

Décision Une décision sélectionne une sortie ou une action parmi plusieurs possibilités.

Ces familles ne doivent pas être confondues. Qualifier n'est pas transporter. Lire n'est pas décider. Décider n'est pas mettre à jour. Mettre à jour n'est pas simplement ajouter.

3.5. La composition

Une transformation isolée est rarement suffisante. Les systèmes réels sont constitués de chaînes :

La composition est notée :

avec :

Mais la simple notation fonctionnelle ne dit pas tout. Lorsque plusieurs transformations sont composées, leurs effets informationnels s'accumulent. Si (f_1) perd l'ordre, aucune transformation ultérieure ne peut le restaurer avec certitude sans information externe. Si (f_2) ajoute une qualification contextuelle, cette qualification peut influencer toutes les étapes suivantes. Si (f_3) agrège plusieurs dimensions en un seul score, les dimensions initiales deviennent potentiellement indéterminables. Le contrat global de composition ne peut donc pas être obtenu par une simple juxtaposition descriptive. Il doit être calculé.

3.6. L'ordre des transformations

Dans certains systèmes, l'ordre n'a pas d'importance. Si deux opérations commutent :

alors leur ordre peut être échangé sans modifier le résultat. Mais de nombreuses opérations linguistiques, cognitives et perceptives ne commutent pas.

Prenons un exemple simple. Appliquer une négation, puis une intensification : pas bon vraiment pas bon n'est pas nécessairement équivalent à : vraiment bon pas vraiment bon Les mêmes éléments sont présents, mais leur ordre d'application modifie la lecture. De même, dans une image :

- flouter puis détecter les contours ;
- détecter les contours puis flouter ;

ne produisent pas le même résultat. Dans un système de décision :

- filtrer puis classer ;
- classer puis filtrer ;

peuvent conduire à des ensembles différents.

La composition ordonnée devient donc un objet central.

3.7. Compose dans l'ÉTR

Compose désigne la construction ordonnée d'un état à partir d'une séquence de transformations. Soit une séquence :

À chaque unité (a_i) est associé un opérateur (A_i). L'état final peut être construit par :

L'ordre conventionnel d'écriture doit être déclaré, car la première opération appliquée est ici (A_1). Si les opérateurs ne commutent pas :

alors Compose conserve intrinsèquement l'ordre de la séquence. C'est important pour le langage. Les phrases : Le chien poursuit le chat. et : Le chat poursuit le chien. contiennent presque les mêmes unités, mais leur ordre modifie les rôles et le sens. Une représentation fondée sur une collection non ordonnée ne suffit donc pas. Compose ne prétend pas, à lui seul, produire le sens complet. Il construit une structure ordonnée qui pourra ensuite être qualifiée. Cette séparation est centrale :

é é

3.8. Structure et qualification

Supposons que Compose produise un état :

Une qualification contextuelle produit ensuite :

où :

- (R) est le référentiel ;
- (\Pi) contient le contexte et la provenance ;
- (\Lambda) est une transformation admissible.

Cette architecture évite d'introduire implicitement le sens dans la construction structurelle. Elle permet aussi de comparer plusieurs qualifications d'une même structure.

On peut alors étudier :

- leurs divergences ;
- leur sensibilité au contexte ;
- leur stabilité ;
- leurs frontières de validité.

Cette distinction est particulièrement utile lorsqu'un mot est polysémique. Le terme « avocat » conserve sa forme structurelle. Sa qualification dépend du contexte :

- avocat comme juriste ;
- avocat comme fruit.

La structure ne doit pas être reconstruite arbitrairement à chaque interprétation. C'est la qualification qui doit déclarer le changement de lecture.

3.9. Pourquoi borner la qualification

Une qualification non contrainte pourrait transformer n'importe quelle structure en n'importe quelle interprétation. Le système deviendrait alors impossible à falsifier. Pour éviter cela, l'ÉTR introduit l'idée d'un domaine admissible de modulation.

où (\mathcal{A}_{\kappa}) décrit l'ensemble des qualifications autorisées
selon une amplitude (\kappa).

Le principe n'impose pas encore une forme unique. Il pose une exigence : Une qualification doit être limitée par un domaine déclaré avant son application. Cette limite peut dépendre :

- du type de donnée ;
- de la confiance du modèle ;
- du contexte disponible ;
- de la tâche ;
- du coût d'une erreur ;
- de la stabilité recherchée.

Dans un système médical, une qualification sémantique incertaine devra être fortement bornée. Dans une application artistique, le domaine admissible pourra être plus large. La même architecture peut donc accueillir plusieurs régimes sans modifier son principe.

3.10. Le chemin comme information

Dans de nombreux systèmes, seul l'état final est conservé.

Mais si plusieurs chemins conduisent à (y), ils ne sont pas nécessairement équivalents.

et :

peuvent produire le même résultat tout en ayant :

- des coûts différents ;
- des pertes différentes ;
- des niveaux de confiance différents ;
- des implications différentes ;
- des degrés de réversibilité différents.

L'ÉTR conserve donc le chemin :

et pas seulement son extrémité. Cette conservation ouvre une possibilité importante : comparer des processus, et non seulement des résultats.

3.11. Chemins fermés et cohérence

Un chemin fermé revient à son point de départ.

Si le système est parfaitement cohérent et réversible selon le contrat annoncé, on peut attendre :

ou, pour des transports multiplicatifs :

où (I) est l'identité. Si ce produit diffère de l'identité, le cycle présente un résidu.

Ce résidu peut signaler :

- une perte ;
- une contradiction ;
- une dépendance au chemin ;
- une qualification contextuelle ;
- une erreur de mesure ;
- une transformation non inversible ;
- une structure non intégrable.

IL NE DOIT PAS ÊTRE INTERPRÉTÉ AUTOMATIQUEMENT COMME UNE ANOMALIE. IL constitue d'abord une observation à expliquer.

3.12. La mémoire comme ensemble structuré de transformations

Une mémoire conventionnelle conserve des données. Une mémoire relationnelle conserve également les transformations qui relient ces données. Nous pouvons représenter une mémoire ÉTR par :

où :

- (V) contient les états ou objets ;
- (E) contient les transformations ou relations ;
- (\Gamma) contient les chemins ;
- (\Pi) contient les provenances ;
- (R) contient les référentiels.

Une mise à jour ne consiste donc pas seulement à ajouter un nouvel élément.

Elle doit préciser :

- si un nouvel objet est créé ;
- si une relation existante est modifiée ;
- si un chemin est prolongé ;
- si un conflit apparaît ;
- si une ancienne lecture devient invalide ;
- si une nouvelle provenance est attachée.

La mémoire devient ainsi un système d'évolution, non un simple stockage.

3.13. Comparaison avec le Transformer

Un Transformer transforme une séquence de tokens en représentations contextuelles en utilisant notamment l'attention. Pour une forme simplifiée :

puis :

Cette opération permet à chaque position de pondérer l'information provenant des autres positions. Le Transformer construit donc des relations contextuelles très riches. Mais son objectif principal n'est pas de déclarer chaque transformation relationnelle comme un objet externe typé et auditable.

Ces relations sont intégrées au calcul du modèle. L'ÉTR adopte un autre niveau d'abstraction. Elle ne cherche pas à remplacer l'attention. Elle cherche à pouvoir déclarer :

- quelle lecture a été produite ;
- dans quel référentiel ;
- par quelle transformation ;
- avec quelles pertes ;
- avec quelle provenance ;
- avant quelle décision.

Un Transformer peut donc constituer l'un des producteurs de qualification ou de lecture d'un système ÉTR.

3.14. Comparaison avec le CNN

Un réseau convolutif applique des noyaux locaux partagés. Pour une convolution 2D simplifiée :

Le noyau (K) détecte des motifs locaux. Les couches successives construisent des représentations de plus en plus abstraites. Le CNN excelle lorsque la structure locale et la régularité spatiale sont importantes. Mais, comme le Transformer, il n'a pas pour objectif premier de produire un registre explicite de toutes les transformations informationnelles. Dans une architecture combinée :

puis :

É

é é

Le CNN extrait. L'ÉTR déclare, relie, compare et préserve. Cette formulation ne hiérarchise pas les deux systèmes. Elle distingue leurs fonctions.

3.15. Ce que cette différence implique

Un modèle neuronal apprend principalement des paramètres lui permettant de produire des représentations ou des sorties utiles. L'ÉTR structure principalement les conditions dans lesquelles des transformations deviennent déclarables, comparables et auditables. On peut résumer ainsi :

è

É é

Les deux approches peuvent être combinées. Le modèle neuronal produit une lecture. L'ÉTR inscrit cette lecture dans un référentiel. Le modèle neuronal estime une qualification. L'ÉTR vérifie son admissibilité. Le modèle neuronal propose une décision. L'ÉTR conserve le chemin, les conditions et la mise à jour qui en résultent.

Conclusion de la première partie

Les trois premiers chapitres ont établi une progression logique. Premièrement, toute intelligence artificielle dépend d'une représentation. Deuxièmement, toute représentation transforme l'objet initial et doit déclarer ses effets informationnels. Troisièmement, une comparaison n'est valide que si les objets ont été rendus comparables dans un référentiel défini. Quatrièmement, les transformations qui construisent cette comparabilité doivent être distinguées des décisions qui en exploitent les résultats. Cinquièmement, l'ordre, les chemins, la provenance et les pertes constituent eux-mêmes des informations. L'ÉTR commence ici, cet endroit précis. Elle ne commence pas par une nouvelle architecture neuronale. Elle commence par une exigence scientifique :

ê é é ê é

La partie suivante examinera les grandes familles d'intelligence artificielle selon cette grille : représentation, transformation, comparaison, décision, mémoire et traçabilité.

DEUXIÈME PARTIE

Comprendre les grandes familles d’intelligence artificielle

Chapitre 4 — Les moteurs de recherche : retrouver avant de comprendre

Avant les grands modèles de langage, avant les réseaux neuronaux profonds et avant les systèmes multimodaux, une question structurait déjà une grande partie de l’informatique : Comment retrouver une information pertinente dans un ensemble beaucoup plus vaste que ce qu’un humain peut parcourir ? Cette question paraît plus simple que la compréhension du langage. Elle ne l’est pas nécessairement. Un moteur de recherche doit accomplir au moins quatre opérations distinctes : . représenter les documents ;

. représenter la requête ;

. comparer la requête aux documents ;

. classer les résultats.

On retrouve déjà la séparation introduite précédemment :

é é

Le moteur de recherche n’a pas nécessairement besoin de « comprendre » les documents au sens humain du terme. Il doit produire un ordre utile entre eux. Cette nuance est essentielle.

4.1. Le problème documentaire

Soit un corpus :

et une requête :

Le moteur cherche à produire un classement :

tel que les premiers documents soient les plus utiles relativement à la requête. Mais le terme « utile » doit être défini. Un document peut être jugé pertinent parce qu'il :

- contient les mêmes mots ;
- traite du même sujet ;
- répond précisément à la question ;
- provient d'une source fiable ;
- est récent ;
- correspond au profil de l'utilisateur ;
- a été fréquemment consulté ;
- satisfait une contrainte juridique ou commerciale.

Ainsi, la pertinence n'est pas une propriété purement intrinsèque du document. Elle est une relation :

où :

- (d) est le document ;
- (q) est la requête ;
- (R) est le référentiel de recherche ;
- (U) représente l'utilisateur ou son profil ;
- (C) représente le contexte.

Le même document peut être très pertinent dans un contexte et inutile dans un autre.

4.2. L'indexation

Parcourir tous les documents à chaque requête serait trop coûteux. Les moteurs construisent donc un index. L'idée la plus simple consiste à associer chaque terme aux documents qui le contiennent. Par exemple :

Cette structure est appelée index inversé. Elle inverse la relation naturelle :

en :

L'indexation est déjà une transformation. Elle conserve :

- la présence de certains termes ;
- leur fréquence ;
- parfois leur position ;
- parfois leur voisinage.

Elle peut perdre ou simplifier :

- l'ordre global du raisonnement ;
- l'ironie ;
- la causalité ;
- la structure argumentative ;
- certaines relations sémantiques.

Le contrat de cette représentation dépend de la sophistication de l'index.

4.3. La fréquence des termes

Une première méthode consiste à compter les occurrences d'un mot dans un document. Soit :

la fréquence du terme (t) dans le document (d). Un terme fréquent dans un document peut être important pour ce document. Mais cette règle produit immédiatement un problème. Des termes comme :

- le ;
- de ;
- un ;
- être ;
- faire ;

apparaissent dans de très nombreux documents sans être discriminants. Il faut donc combiner la fréquence locale avec la rareté globale. On introduit alors :

où :

- (N) est le nombre total de documents ;
- (\operatorname{df}(t)) est le nombre de documents contenant (t).

Le produit :

attribue davantage de poids aux termes fréquents dans un document, mais rares dans le corpus. Cette méthode a joué un rôle fondamental dans la recherche documentaire. Elle ne « comprend » pas le document. Elle construit une représentation discriminante.

4.4. Le modèle vectoriel documentaire

Un document peut être représenté par un vecteur :

où chaque composante correspond à un terme du vocabulaire. La requête devient elle aussi un vecteur :

La similarité peut alors être calculée avec le cosinus :

Cette méthode permet de comparer les orientations plutôt que les seules magnitudes. Deux documents de longueurs différentes peuvent être considérés

comme proches s'ils distribuent leurs termes de manière similaire. Le moteur produit alors :

puis trie les documents selon (s_i). Cette architecture est claire :

Elle illustre parfaitement la distinction entre lecture, comparaison et décision.

4.5. BM25 : la pertinence comme fonction calibrée

Les moteurs documentaires modernes (Base théorique 1976 - Okapi BM25) utilisent souvent des fonctions plus élaborées, comme BM25. Une forme simplifiée s'écrit :

$$\frac{\qquad\qquad\qquad\qquad\qquad\qquad}{\quad}$$

où :

- (f(t,d)) est la fréquence du terme dans le document ;
- (| d |) est la longueur du document ;
- (\operatorname{avgdl}) est la longueur moyenne du corpus ;
- (k_1) contrôle la saturation de fréquence ;
- (b) contrôle la normalisation par la longueur.

BM25 montre qu'un bon système ne se contente pas d'ajouter des occurrences. Il introduit des hypothèses :

- répéter un terme n'augmente pas indéfiniment sa pertinence ;
- les documents longs doivent être normalisés ;
- les termes rares doivent compter davantage.

Ces choix ne sont pas neutres. Ils constituent un modèle de pertinence.

4.6. Ce que le moteur documentaire sait réellement

Un moteur documentaire classique sait très bien :

- indexer ;
- filtrer ;
- compter ;
- classer ;
- retrouver ;
- normaliser ;
- combiner plusieurs signaux.

Il peut aussi intégrer :

- l'autorité d'une source ;
- la popularité ;
- la récence ;
- la localisation ;
- la personnalisation ;
- des règles métier.

Il ne faut donc pas réduire la recherche documentaire à un simple comptage de mots. Cependant, même lorsqu'il est sophistiqué, le moteur poursuit principalement un objectif :

Il ne cherche pas nécessairement à reconstruire un chemin relationnel complet entre la requête et chaque document.

4.7. La différence entre retrouver et expliquer

Supposons qu'un moteur place un document en première position. Il peut parfois expliquer ce résultat par :

- présence des termes de la requête ;
- forte autorité du domaine ;
- adéquation géographique ;
- récence ;
- historique de consultation.

Mais cette explication reste souvent agrégée. Le moteur ne décrit pas nécessairement :

- quelles transformations sémantiques ont rendu le document

comparable ;

- quelles dimensions ont été sacrifiées ;
- quelles alternatives de lecture existaient ;
- quelles relations internes du document ont produit sa pertinence ;
- pourquoi deux documents proches ont reçu des rangs différents au-

delà du score final.

L'ÉTR ne remplace pas le moteur de recherche. Elle peut lui ajouter une couche de déclaration. Par exemple :

avec, pour chaque résultat :

é

Le classement devient alors accompagné d'un dossier relationnel.

4.8. Exemple : recherche sur un terme ambigu

Considérons la requête : avocat responsabilité civile

Le terme « avocat » peut désigner :

- un professionnel du droit ;
- un fruit.

Le moteur documentaire exploite le contexte « responsabilité civile » pour favoriser l'interprétation juridique. Cette désambiguïsation peut être produite par :

- cooccurrence ;
- statistiques du corpus ;
- modèle sémantique ;
- règles ;
- embeddings.

L'ÉTR demanderait que cette transformation soit déclarée comme qualification :

é

avec :

- le contexte utilisé ;
- le référentiel juridique ;
- le niveau de confiance ;
- les interprétations écartées ;
- la possibilité d'abstention.

Ainsi, la désambiguïsation ne disparaît pas dans le score final. Elle reste une opération inspectable.

4.9. La recherche comme pipeline relationnel

Un moteur enrichi par l'ÉTR pourrait être décrit ainsi :

ê

é é é

à é

Chaque étape possède son propre contrat. Cela permettrait de distinguer plusieurs causes d'échec. Un mauvais résultat peut provenir :

- d'une mauvaise segmentation ;
- d'une mauvaise qualification ;
- d'un référentiel inadéquat ;
- d'une métrique mal choisie ;
- d'une règle de classement ;
- d'une mémoire utilisateur erronée.

Sans séparation, ces erreurs sont souvent confondues dans une même « mauvaise réponse ».

4.10. Ce que l'ÉTR apporte à la recherche documentaire

L'apport potentiel n'est pas d'améliorer automatiquement tous les classements. Il est de rendre le processus plus structuré. L'ÉTR peut permettre de :

- conserver la provenance des qualifications ;
- distinguer proximité lexicale et proximité conceptuelle ;
- représenter plusieurs types de relations simultanément ;
- comparer les chemins de recherche ;
- mémoriser les bifurcations ;
- déclarer les transformations ayant conduit au classement ;
- identifier les pertes informationnelles ;
- faire évoluer le référentiel sans écraser l'historique.

Ainsi, le moteur documentaire continue de retrouver. L'ÉTR ajoute la possibilité de décrire comment la recherche a été construite.

Chapitre 5 — Les embeddings : construire un espace de proximité

Les embeddings ont transformé la recherche, la recommandation et le traitement du langage. Leur principe général est puissant : représenter des objets complexes par des vecteurs dans un espace où certaines relations deviennent géométriquement calculables. Un mot, une phrase, une image ou un utilisateur peuvent être associés à un vecteur :

La fonction :

est appelée fonction d'encodage. Le but n'est pas nécessairement de conserver toutes les propriétés de (x). Le but est de conserver celles qui sont utiles à la tâche.

5.1. De la représentation symbolique à la représentation distribuée

Dans une représentation symbolique classique, un terme possède une identité distincte. Par exemple :

Dans un encodage one-hot, chaque mot correspond à une dimension différente. Si le vocabulaire contient quatre termes :

on peut écrire :

Cette représentation conserve parfaitement l'identité. Mais elle ne représente aucune proximité. La distance entre « chat » et « chien » est la même que celle entre « chat » et « voiture ». Les embeddings modifient cela. Ils apprennent des représentations distribuées :

Les vecteurs proches peuvent alors correspondre à des objets sémantiquement proches.

5.2. Le principe distributionnel

Une intuition historique importante du traitement du langage est la suivante : des mots apparaissant dans des contextes similaires tendent à partager certaines propriétés sémantiques. Un mot n'est donc plus défini uniquement par une entrée lexicale. Il est aussi défini par les relations statistiques de son usage.

Soit (w) un mot et (C(w)) son ensemble de contextes observés. Une représentation peut être apprise comme :

où (F) condense les régularités contextuelles. Le sens n'est pas explicitement écrit dans chaque composante. Il est distribué dans la géométrie de l'espace.

5.3. La proximité géométrique

Deux objets peuvent être comparés par distance euclidienne :

ou par similarité cosinus :

Une valeur élevée indique une orientation proche. Cette approche permet de retrouver des objets similaires sans exiger qu'ils partagent les mêmes mots. Par exemple : défaillance du moteur

électrique peut être rapproché de : panne du système de propulsion même si les formulations diffèrent. C'est un progrès majeur par rapport à une recherche purement lexicale.

5.4. Une proximité n'est pas une identité

Deux vecteurs proches ne signifient pas nécessairement que les objets sont équivalents. Ils peuvent partager :

- un sujet ;
- un registre ;
- un contexte ;
- une fonction ;
- une tonalité ;
- une structure.

Mais ils peuvent aussi diverger sur un point essentiel. Reprenons : Le médicament réduit le risque. Le médicament ne réduit pas le risque. Leur proximité vectorielle peut rester élevée parce que presque tout leur contexte est identique. Pourtant, leur valeur propositionnelle est opposée. Ainsi :

é é

n'implique pas :

La similarité est une relation parmi d'autres.

5.5. La compression sémantique

Un embedding compresse un objet potentiellement très riche dans un vecteur de dimension finie. Soit un document contenant plusieurs milliers de mots. Il peut devenir :

ou dans un autre espace de dimension différente. Cette compression rend la recherche extrêmement rapide. Mais elle implique une agrégation. Des propriétés multiples sont condensées dans une même représentation. Il devient difficile de savoir précisément :

- quelle phrase a déterminé la proximité ;
- quel argument a été conservé ;
- quelle nuance a été perdue ;
- quelle contradiction interne a été lissée ;
- quel passage a produit le score.

Les embeddings sont donc très performants pour la récupération approximative, mais moins naturellement adaptés à une explication relationnelle détaillée.

5.6. L'espace n'est pas neutre

L'espace vectoriel dépend du modèle et de son apprentissage. Deux modèles peuvent associer au même objet deux représentations différentes :

et produire des voisinages différents.

Le référentiel implicite dépend donc :

- du corpus d'apprentissage ;
- de l'objectif ;
- de l'architecture ;
- de la langue ;
- de la fonction de perte ;
- du protocole d'optimisation.

L'espace vectoriel constitue lui-même un référentiel appris. Mais ce référentiel n'est pas toujours déclaré de manière explicite dans l'usage final.

5.7. Les dimensions interprétables et latentes

Dans certains espaces, chaque dimension possède une signification claire. Par exemple :

é é

Dans un embedding appris, les dimensions sont généralement latentes. La composante (x_{317}) ne signifie pas directement « négation », « fruit » ou « droit ». Le sens est distribué entre plusieurs dimensions et plusieurs interactions. Cette distribution augmente la puissance de représentation. Elle diminue l'interprétabilité directe. Le système peut donc savoir distinguer des objets sans pouvoir fournir une définition simple de chaque axe.

5.8. Les analogies vectorielles

Certains espaces d'embeddings permettent des opérations analogiques. On a popularisé des relations de type :

L'intérêt de cet exemple n'est pas sa perfection, mais l'idée qu'une transformation sémantique puisse être représentée par une direction.

Cette direction peut être interprétée comme une transformation approximative dans l'espace. L'ÉTR rencontre ici un point de contact important. Mais elle pose une question supplémentaire : Cette direction constitue-t-elle une transformation déclarée, stable, typée et transférable, ou seulement une régularité géométrique observée ?

Une analogie vectorielle peut être utile sans constituer une loi générale.

5.9. L'embedding comme lecture

Dans le langage de l'ÉTR, un embedding peut être considéré comme une lecture :

Cette lecture possède un contrat. Elle conserve potentiellement :

- certaines proximités sémantiques ;
- certaines régularités d'usage ;
- certaines structures contextuelles.

Elle transforme :

- le texte en géométrie ;
- la pluralité des propriétés en coordonnées distribuées.

Elle peut rendre indéterminables :

- la formulation exacte ;
- l'ordre intégral ;
- la provenance de chaque nuance ;
- certaines contradictions locales.

Cette description ne critique pas l'embedding. Elle explicite sa fonction.

5.10. L'embedding comme référentiel partiel

L'espace d'embedding peut aussi être vu comme un référentiel partiel :

où :

- (\mathbb{R}^{d}) est l'espace ;
- (\rho_e) est la métrique ;
- (\mathcal{N}) est la structure de voisinage ;
- (\Pi_e) est la provenance du modèle.

Mais ce référentiel ne contient pas nécessairement :

- une ontologie explicite ;
- des types de relations distincts ;
- une frontière sémantique claire ;
- un registre des transformations ;
- un historique des qualifications.

L'ÉTR peut donc encapsuler l'espace vectoriel sans s'y réduire.

5.11. Recherche vectorielle et approximation

Lorsque la base contient des millions de vecteurs, comparer la requête à tous les objets devient coûteux. On utilise alors des structures de recherche approximative des plus proches voisins. L'objectif est de trouver rapidement :

où ANN signifie approximate nearest neighbors. Cette approximation est un choix calculatoire. Elle peut introduire une différence entre :

- le voisin théoriquement le plus proche ;
- le voisin effectivement retrouvé.

Dans un système ÉTR, cette différence devrait être déclarée. Le pipeline devient :

é

avec :

- approximation ;
- paramètres d'index ;
- taux de rappel attendu ;
- métrique ;
- seuils.

La recherche vectorielle n'est donc pas une opération unique, mais une chaîne de transformations.

5.12. Les limites structurelles du score unique

Supposons qu'un système compare deux objets selon plusieurs dimensions :

é é

é

é

é

Un score global pourrait être :

Mais cette agrégation peut masquer des profils très différents. Deux documents peuvent obtenir le même score :

tout en ayant :

L'un peut être très proche thématiquement mais peu fiable. L'autre peut être plus distant, mais parfaitement cohérent et documenté. Le score unique produit un ordre. Il ne conserve pas nécessairement la structure relationnelle.

5.13. La relation comme profil plutôt que comme nombre

L'ÉTR propose de conserver un profil relationnel :

avant toute agrégation. La décision peut ensuite dépendre de la tâche. Pour une recherche exploratoire :

favorisera la diversité. Pour une recherche juridique :

favorisera la fiabilité, la précision et la temporalité normative. Pour une recommandation culturelle :

pourra accepter davantage de distance. Le même profil relationnel peut donc être routé différemment. Cette séparation évite d'inscrire une politique de décision dans la représentation elle-même.

5.14. L'embedding et la mémoire

Une base vectorielle conserve souvent :

- un identifiant ;
- un vecteur ;
- des métadonnées ;
- parfois le contenu source.

Cette structure est très efficace. Mais elle ne conserve pas nécessairement :

- pourquoi le vecteur a été produit ;
- avec quelle version du modèle ;
- quelles transformations ont précédé l'encodage ;
- quelles relations ont changé lors d'une mise à jour ;
- comment le voisinage a évolué.

L'ÉTR peut ajouter une mémoire d'évolution :

avec :

mais surtout avec le contexte de transformation :

è è

La variation géométrique devient alors documentée.

5.15. Compatibilité entre embeddings et ÉTR

Les embeddings constituent probablement l'un des composants les plus immédiatement compatibles avec l'ÉTR. Ils peuvent servir à :

- proposer des candidats ;
- estimer une proximité ;
- produire une qualification initiale ;
- détecter des clusters ;
- construire des ponts sémantiques ;
- réduire l'espace de recherche.

L'ÉTR peut ensuite :

- typer les relations ;
- conserver les dimensions ;
- ajouter la provenance ;
- distinguer lecture et décision ;
- appliquer des contraintes ;
- mémoriser les chemins.

L'architecture devient :

é

É

é é

é ç

5.16. Ce que l’embedding apporte et ce qu’il ne déclare pas

L’embedding apporte :

- une géométrie compacte ;
- une proximité calculable ;
- une forte capacité de généralisation ;
- une compatibilité multimodale ;
- une recherche rapide.

Il ne déclare pas nécessairement :

- la nature de chaque relation ;
- le mécanisme exact d’une transformation ;
- les pertes par dimension ;
- les conditions de comparabilité ;
- le chemin ayant produit la décision ;
- la validité hors du corpus d’apprentissage.

L’ÉTR n’abolit pas cette puissance. Elle cherche à la rendre composable dans un système plus explicite.

Chapitre 6 — Les réseaux convolutifs : apprendre par motifs locaux

Les réseaux convolutifs, ou CNN, ont joué un rôle déterminant dans le développement de la vision artificielle moderne. Leur force provient d’une hypothèse simple et puissante : les motifs locaux sont importants, et un même motif peut apparaître à plusieurs positions. Un bord vertical reste un bord vertical qu’il apparaisse à gauche ou à droite de l’image. Une texture peut se répéter. Une forme locale peut contribuer à la reconnaissance d’un objet global. Le CNN exploite cette régularité spatiale.

6.1. L’image comme tenseur

Une image numérique peut être représentée par :

où :

- (H) est la hauteur ;
- (W) est la largeur ;
- (C) est le nombre de canaux.

Pour une image RGB :

Chaque pixel possède donc trois composantes principales. Le CNN ne reçoit pas directement « un visage » ou « une voiture ». Il reçoit une organisation de valeurs numériques.

6.2. Le noyau convolutif

Un noyau, ou filtre, est un petit tenseur de poids. Pour simplifier :

Il est déplacé sur l’image. À chaque position (p), il calcule une combinaison locale :

Le même noyau est partagé spatialement.

Cette propriété s’appelle partage de poids. Elle réduit fortement le nombre de paramètres et introduit une forme d’équivariance à la translation.

6.3. Détecter sans nommer

Un filtre peut répondre fortement à certains motifs :

- contours ;
- orientations ;
- textures ;
- transitions de luminosité.

Mais le filtre ne porte pas nécessairement un nom sémantique explicite. Il apprend une fonction utile à l'optimisation globale. Dans les premières couches, certaines activations sont relativement interprétables. Dans les couches profondes, les représentations deviennent plus composites. Le réseau peut apprendre à reconnaître un visage sans posséder une règle symbolique simple : si deux yeux, un nez et une bouche, alors visage. La décision émerge de nombreuses transformations successives.

6.4. Hiérarchie des représentations

Un CNN profond construit plusieurs niveaux :

Les premières couches détectent souvent des structures locales simples. Les couches intermédiaires combinent ces motifs. Les couches profondes peuvent répondre à des formes plus abstraites. Cette hiérarchie est l'une des grandes forces du CNN. Elle permet au réseau d'apprendre les caractéristiques plutôt que de les coder entièrement à la main.

6.5. Convolution et localité

La convolution impose un voisinage local. Une unité de sortie dépend d'une fenêtre limitée de l'entrée. Après plusieurs couches, le champ réceptif s'élargit. Ainsi, une activation profonde peut dépendre d'une région importante de l'image. Mais cette dépendance globale est construite progressivement.

é

Cette progression correspond bien aux structures spatiales. Elle est moins naturellement adaptée à certaines dépendances longues ou non locales, ce qui a motivé d'autres architectures.

6.6. Pooling et réduction

Les CNN utilisent souvent des opérations de réduction spatiale. Par exemple, le max-pooling :

conserve l'activation maximale dans une région. Cette opération réduit la résolution. Elle peut améliorer la robustesse aux petites translations. Mais elle supprime une partie de l'information de position précise. Dans le contrat de transformation :

- l'intensité locale dominante est conservée ;
- certains détails fins sont supprimés ;
- la position exacte peut devenir indéterminable.

Le pooling est donc un exemple clair de transformation utile mais non neutre.

6.7. Invariance et équivariance

Deux notions doivent être distinguées. Équivariance Une transformation de l'entrée produit une transformation correspondante de la sortie.

Pour une translation (\tau) :

Invariance La sortie reste identique malgré une transformation de l'entrée.

Les couches convolutives sont principalement équivariantes aux translations dans certaines conditions. Les opérations de pooling et d'agrégation contribuent à une invariance partielle. Cette distinction montre que le réseau ne « supprime » pas immédiatement la position. Il transforme progressivement la manière dont elle compte.

6.8. Le CNN comme composition de transformations

Un CNN peut être représenté par :

Chaque couche applique :

où :

- (K_l) est l'ensemble des noyaux ;
- (*) désigne la convolution ;
- (b_l) est un biais ;
- (\sigma) est une fonction d'activation.

Le réseau entier est une composition ordonnée.

L'ordre des couches est déterminant.

dans le cas général. Cela rejoint directement la question de Compose dans l'ÉTR. Mais le CNN compose des transformations apprises dans un but prédictif. L'ÉTR cherche à déclarer leur rôle relationnel, leurs contrats et leurs effets.

6.9. Ce que le CNN conserve

Selon son architecture, un CNN peut conserver ou exploiter :

- la structure locale ;
- les motifs répétés ;
- les relations spatiales ;
- les hiérarchies de formes ;
- certaines invariances.

Il peut transformer ou réduire :

- la résolution ;
- la localisation exacte ;
- la texture ;
- certaines amplitudes ;
- des détails jugés non utiles à la tâche.

Le réseau n'est pas « infidèle » à l'image. Il construit une représentation orientée vers son objectif.

6.10. Une activation n'est pas une relation explicite

Supposons qu'un réseau reconnaisse un animal. Certaines activations ont contribué à la décision. Mais le réseau ne produit pas nécessairement une structure explicite comme :

é

Il peut être possible de produire des cartes d'activation ou des méthodes d'explication. Mais celles-ci sont des lectures supplémentaires du modèle. Elles ne sont pas toujours équivalentes à un registre natif de transformations déclarées. Cette différence ne diminue pas la performance du CNN. Elle précise son architecture.

6.11. Comparaison entre CNN et moteur documentaire

Le moteur documentaire traite principalement des occurrences et des relations lexicales ou sémantiques entre documents. Le CNN traite principalement des structures locales organisées dans l'espace. Le moteur documentaire demande : Quels documents correspondent à cette requête ? Le CNN demande : Quels motifs permettent de produire la sortie attendue ? Le premier construit un

classement. Le second apprend une fonction de représentation et de décision. Leur notion de proximité n'est pas la même.

6.12. Le noyau fixe et le contexte

Dans une convolution classique, le noyau appris reste généralement identique pour toutes les positions :

Cela signifie que la règle locale est partagée. Cette propriété est efficace lorsque le même motif doit être reconnu partout. Mais certaines situations exigent une adaptation à la position ou au contexte. C'est précisément l'une des motivations des réseaux involutifs et des mécanismes dynamiques.

6.13. CNN et ÉTR

Un CNN peut produire une lecture visuelle :

L'ÉTR peut ensuite traiter cet état comme un objet typé. Par exemple :

é

é

é

Le CNN reste responsable de l'extraction perceptive. L'ÉTR devient responsable de la déclaration relationnelle. Cette séparation peut être particulièrement utile en robotique, en médecine ou dans les systèmes multimodaux.

6.14. Exemple : inspection industrielle

Une caméra observe une pièce mécanique. Le CNN détecte :

- une fissure ;
- une variation de surface ;
- une anomalie de forme.

Il produit :

L'ÉTR peut ensuite relier cette lecture à :

- la référence de la pièce ;
- son historique de maintenance ;
- les défauts connus ;
- les seuils de sécurité ;
- le coût d'une erreur ;
- le protocole de décision.

La chaîne devient :

é é é

Le CNN ne doit pas être remplacé. Il doit être inséré dans un système où ses résultats deviennent explicites et contextualisés.

6.15. Limites importantes

Un CNN peut être sensible :

- aux changements de domaine ;
- aux perturbations adversariales ;
- aux biais du corpus ;
- aux textures non représentatives ;
- aux variations non rencontrées durant l'apprentissage.

Une forte précision moyenne ne garantit pas une robustesse universelle. Le référentiel d'apprentissage définit en partie le monde dans lequel le modèle est compétent.

L'ÉTR peut contribuer à déclarer cette frontière.

é é

Une prédiction hors de cette frontière doit pouvoir être qualifiée comme incertaine ou inadmissible.

6.16. Ce que le CNN apporte et ce que l'ÉTR ajoute

Le CNN apporte :

- l'apprentissage de motifs ;
- la hiérarchie perceptive ;
- la robustesse locale ;
- la réduction paramétrique ;
- l'efficacité en vision.

L'ÉTR ajoute potentiellement :

- le typage des sorties ;
- la provenance ;
- les contrats de transformation ;
- la séparation entre lecture et décision ;
- la comparaison multi-référentielle ;
- la mémoire des chemins ;
- la déclaration des pertes.

La combinaison devient plus puissante que l'opposition.

Transition Les moteurs de recherche ont montré comment retrouver. Les embeddings ont montré comment construire une géométrie de proximité. Les CNN ont montré comment apprendre des motifs locaux et les composer hiérarchiquement.

Le chapitre suivant introduira une architecture située à l'intersection de plusieurs intuitions :

- le noyau ne sera plus nécessairement identique partout ;
- la transformation dépendra davantage du contexte local ;
- la différence entre opérateur fixe et opérateur généré deviendra

centrale.

Cette architecture est celle des réseaux involutifs. Elle permettra ensuite d'aborder plus rigoureusement les Transformers, où la relation contextuelle ne constitue plus un ajout périphérique, mais le mécanisme principal du calcul.

Chapitre 7 — Les réseaux involutifs : lorsque le noyau dépend du contexte

Les réseaux convolutifs reposent sur un principe de partage spatial :

Le même noyau est appliqué à plusieurs positions. Cette stabilité constitue l'une de leurs forces. Elle réduit le nombre de paramètres et permet de reconnaître un même motif dans différentes régions de l'image. Mais elle impose également une hypothèse : la règle de traitement locale peut rester identique quel que soit le contenu rencontré à cette position. Cette hypothèse est souvent utile. Elle n'est pas toujours suffisante. Une texture, une forme ou une structure locale peut nécessiter un traitement différent selon :

- sa position ;
- son voisinage ;
- l'échelle ;
- le contexte global ;
- le canal observé ;
- l'état courant du système.

Les réseaux involutifs introduisent précisément cette variation. Au lieu d'appliquer un noyau spatialement partagé, ils construisent un noyau qui dépend de la position ou du contenu local.

Le traitement n'est plus seulement :

è

mais :

è é

Cette différence est importante pour comprendre l'évolution des architectures neuronales modernes.

7.1. Le terme « involution »

Le mot involution possède un sens mathématique classique. Une fonction (f) est involutive lorsqu'elle est sa propre inverse :

Mais le terme utilisé dans certaines architectures de vision ne désigne pas nécessairement une involution au sens strict de l'algèbre. Il indique plutôt une inversion partielle du principe de convolution. Dans une convolution :

- le noyau varie selon les canaux de sortie ;
- il est partagé spatialement.

Dans une involution architecturale :

- le noyau peut être partagé entre certains canaux ;
- il varie spatialement selon la position ou le contenu.

Il faut donc éviter une confusion de vocabulaire.

Le nom décrit une inversion de conception, non une propriété générale d'inversibilité.

7.2. Convolution : noyau fixe relativement à la position

Pour une convolution simplifiée :

où :

- (p) est la position ;
- (\delta) parcourt le voisinage ;
- (i) désigne les canaux d'entrée ;
- (o) les canaux de sortie ;
- (K) est le noyau appris.

Le noyau ne dépend pas de (p).

Le même motif est donc recherché dans tout l'espace.

7.3. Involution : noyau produit à la position

Dans une forme simplifiée d'involution :

Le noyau (H_{p,\delta}) dépend de la position (p), souvent par

l'intermédiaire de l'entrée locale.

où (G) est une fonction génératrice de noyau. Le calcul complet devient :

puis :

Il y a donc deux transformations successives : . production de l'opérateur ;

. application de l'opérateur.

Cette distinction rapproche l'involution d'une logique méta-opérationnelle. Le système ne choisit plus seulement une sortie. Il construit localement la transformation qui produira cette sortie.

7.4. Opérateur fixe et opérateur conditionnel

Nous pouvons distinguer deux formes. Opérateur fixe

Le paramètre (K) est appris, puis réutilisé. Opérateur conditionnel

L'opérateur dépend d'un état (x). Le calcul devient :

Cette structure est plus expressive.

Mais elle est également plus difficile à auditer. Car la transformation appliquée n'est plus uniquement déterminée par les paramètres globaux du modèle. Elle dépend aussi de l'entrée. Deux positions peuvent subir deux transformations différentes :

même lorsqu'elles appartiennent à la même couche.

7.5. Pourquoi générer un noyau dynamique ?

Un noyau dynamique peut s'adapter à plusieurs situations. Variation de forme Une même catégorie d'objet peut présenter des structures locales très différentes. Variation de contexte Un motif peut avoir une signification différente selon son voisinage. Variation de position Certaines régions d'une image peuvent nécessiter une attention ou une résolution différente. Variation de contenu Le

système peut privilégier certains voisins selon l'information locale. Le noyau devient ainsi un mécanisme de sélection contextuelle.

7.6. Exemple intuitif

Supposons qu'une image contienne :

- un ciel uniforme ;
- un visage ;
- un texte imprimé ;
- une texture végétale.

Une convolution classique applique les mêmes filtres à toutes les régions. Les activations diffèrent, mais le mécanisme local reste partagé. Une architecture involutive peut produire :

- un noyau lissant dans le ciel ;
- un noyau sensible aux contours fins sur le visage ;
- un noyau orienté sur les caractères ;
- un noyau textural dans la végétation.

Le traitement local dépend alors du contenu rencontré. Cette adaptabilité est proche de ce qu'un humain pourrait décrire comme : ne pas regarder toutes les régions de la même manière.

7.7. Le noyau comme sortie intermédiaire

Dans un CNN, le noyau est un paramètre du modèle. Dans une involution, le noyau local devient une sortie intermédiaire.

Le noyau appartient donc à un niveau différent de la donnée. Il est à la fois :

- produit par le réseau ;
- utilisé par le réseau ;
- dépendant du contexte ;
- responsable de la transformation suivante.

Cette structure mérite d'être explicitement distinguée.

Cette proposition dépasse le cas des réseaux involutifs. Elle apparaît également dans :

- les hyperréseaux ;
- les filtres dynamiques ;
- les mécanismes d'attention ;
- les contrôleurs d'agents ;
- les systèmes générant du code ;
- les architectures méta-apprenantes.

7.8. Lecture ÉTR d'un opérateur généré

Dans un cadre ÉTR, une transformation générée pourrait être décrite en deux niveaux.

Niveau 1 — Génération

Niveau 2 — Application

Le contrat complet doit donc déclarer :

- les conditions ayant produit (T_Q) ;
- le domaine de validité de (T_Q) ;
- les bornes de ses paramètres ;
- les données auxquelles il est appliqué ;
- les pertes produites ;
- sa stabilité ;
- sa réutilisabilité éventuelle.

On ne doit pas confondre :

qui génère l'opérateur, avec :

qui transforme l'état.

7.9. La question de l'identité de la transformation

Lorsque l'opérateur dépend de l'entrée, une question apparaît : s'agit-il encore de la même transformation ? Si l'on note simplement :

on masque la dépendance contextuelle. Une notation plus rigoureuse serait :

ou :

Deux applications appartiennent à la même famille, mais ne sont pas nécessairement le même opérateur.

Cette distinction est cruciale pour la comparaison expérimentale. On ne peut pas conclure qu'une couche « applique le même mécanisme » si le mécanisme est généré différemment à chaque entrée.

7.10. Stabilité et variation

La puissance d'un opérateur dynamique vient de sa variation. Sa fragilité potentielle vient également de cette variation. Si une faible modification de l'entrée provoque une forte variation du noyau :

mais :

alors le système peut devenir instable. Il faut donc étudier :

- la continuité ;
- la sensibilité ;
- la borne de variation ;
- la robustesse ;
- la dérive hors distribution.

Une architecture dynamique ne doit pas seulement être expressive. Elle doit être contrôlable.

7.11. Domaine admissible des opérateurs dynamiques

L'ÉTR propose ici une discipline naturelle. L'opérateur produit doit appartenir à un domaine admissible :

La fonction génératrice ne peut donc pas produire n'importe quelle transformation.

Le domaine (\mathcal{A}) peut imposer :

- une norme maximale ;
- une condition de positivité ;
- une borne spectrale ;
- une contrainte d'inversibilité ;
- une conservation d'énergie ;
- une limitation de variation ;
- un ensemble fini de formes autorisées.

La dynamique reste alors flexible, mais non arbitraire.

7.12. Involution et qualification

Une involution architecturale peut être rapprochée d'une qualification locale. L'état (X(p)) produit un opérateur qui module la lecture de son voisinage.

Cependant, cette analogie doit rester précise. Dans l'ÉTR, une qualification est une famille déclarée de transformations contextuelles. Dans une architecture neuronale, le noyau dynamique est appris selon une fonction de perte. Les deux peuvent être compatibles, mais ils ne sont pas identiques par définition. L'ÉTR fournit le langage de description. L'involution fournit une implémentation possible d'une modulation locale.

7.13. Comparaison avec Compose

Compose conserve l'ordre d'une séquence de transformations. L'involution modifie la transformation selon l'état courant. Nous pouvons combiner les deux :

où l'opérateur dépend :

- de l'état courant (Q_n) ;
- de l'unité (a_n) ;
- éventuellement du référentiel (R).

La chaîne devient :

Nous ne composons plus une suite d'opérateurs fixes. Nous composons une suite d'opérateurs générés en fonction de l'histoire

du calcul. Cette structure est plus proche de nombreux phénomènes linguistiques. Le sens d'un mot dépend de l'état construit par les mots précédents.

7.14. Dépendance au chemin

Lorsque chaque opérateur dépend de l'état courant, le chemin devient déterminant. Supposons :

puis :

L'opérateur appliqué à (b) dépend de la transformation précédente. Si l'ordre est inversé :

puis :

alors :

dans le cas général. La non-commutativité ne provient plus seulement des matrices. Elle provient aussi de la génération contextuelle des opérateurs.

7.15. Le risque d'un système sans frontière

Si toute entrée peut produire librement son propre opérateur, le système peut devenir difficile à falsifier. Toute erreur peut être expliquée après coup par une « adaptation contextuelle ». Pour conserver une valeur scientifique, il faut donc déclarer :

- les variables contextuelles autorisées ;
- le domaine de génération ;
- les bornes ;
- les invariants ;
- les cas d'abstention ;
- les critères d'échec.

Un opérateur contextuel n'est pas scientifique parce qu'il s'adapte. Il devient scientifique lorsqu'on peut mesurer les conditions et les limites de cette adaptation.

7.16. Comparaison structurée

Propriété	Convolution	Involution
Noyau	appris et partagé	généré selon la position ou le contenu
Dépendance spatiale	faible ou indirecte	explicite
Partage	spatial	souvent entre groupes de canaux
Adaptation locale	par activation	par génération de noyau
Traçabilité	noyau global inspectable	noyaux locaux nombreux et variables
Risque principal	rigidité contextuelle	instabilité ou suradaptation
Lecture ÉTR	transport local fixe	transformation

conditionnelle

7.17. Ce que les réseaux involutifs apportent

Ils apportent :

- une adaptation locale ;
- une meilleure prise en compte du contenu ;
- une séparation entre génération et application de noyau ;
- une alternative au partage spatial strict ;
- une expressivité accrue.

Ils rendent également plus visible une question générale : la règle doit-elle être fixe, ou doit-elle être produite par la situation ? Cette question sera centrale dans le chapitre suivant.

Les Transformers ne produisent pas exactement des noyaux convolutifs dynamiques. Mais ils déterminent, à chaque entrée, quelles positions doivent influencer quelles autres positions. Le contexte ne modifie plus seulement un filtre local. Il modifie la structure relationnelle du calcul.

Chapitre 8 — Les Transformers : construire une représentation par relations contextuelles

Les Transformers constituent aujourd'hui l'architecture dominante des grands modèles de langage. Ils sont également utilisés en vision, en audio, en biologie, en robotique et dans de nombreuses tâches multimodales. Leur succès repose sur plusieurs propriétés :

- traitement efficace des séquences ;
- modélisation des dépendances longues ;
- parallélisation de l'apprentissage ;
- représentations contextuelles ;
- capacité de montée en échelle ;
- compatibilité avec l'apprentissage auto-supervisé.

Mais leur mécanisme central est souvent simplifié à l'excès. On dit parfois : le Transformer comprend les mots grâce à l'attention. Cette formulation est insuffisante. Un Transformer ne reçoit pas directement des mots, ni des concepts déjà constitués. Il reçoit des tokens transformés en vecteurs, puis modifie ces vecteurs à travers plusieurs couches. L'attention ne constitue pas une interprétation humaine. Elle est une opération de pondération et d'agrégation contextuelle.

8.1. Du texte aux tokens

Une phrase : Le chat dort sur le canapé. est d'abord transformée en une séquence de tokens.

Un token peut correspondre à :

- un mot ;
- une partie de mot ;
- un signe de ponctuation ;
- un caractère ;
- un fragment fréquent.

Le découpage dépend du tokenizer. Deux modèles peuvent tokeniser la même phrase différemment.

La tokenisation constitue donc une première transformation structurante. Elle influence :

- la longueur de la séquence ;
- la granularité ;
- le coût du calcul ;
- les unités visibles par le modèle ;
- certaines difficultés linguistiques.

8.2. Tokenisation et perte

La tokenisation est souvent largement réversible. Mais elle n'est pas neutre. Elle peut fragmenter certains mots de manière inhabituelle. Elle peut traiter différemment :

- les langues ;
- les noms rares ;
- les symboles ;
- le code ;
- les chiffres ;
- les écritures non latines.

Le contrat de tokenisation doit donc déclarer :

- le vocabulaire ;
- la méthode ;
- les règles de normalisation ;
- les marqueurs spéciaux ;
- la reconstruction possible ;
- les cas ambigus.

Dans une lecture ÉTR :

8.3. De l'identité du token au vecteur

Chaque token reçoit une représentation vectorielle initiale.

où (E) est une matrice d'embedding. On obtient :

À ce stade, le même token possède généralement la même représentation initiale. Mais son interprétation finale dépendra du contexte. Le mot « avocat » commence par un vecteur lexical commun.

Ses représentations internes divergeront ensuite selon la phrase.

8.4. La position

L'attention seule ne connaît pas nécessairement l'ordre. Il faut donc introduire une information positionnelle. On peut écrire :

où (p_i) encode la position. D'autres architectures utilisent des encodages positionnels relatifs ou rotatifs. La fonction est la même au niveau conceptuel : permettre au modèle de distinguer les positions et leurs relations. Sans cette information, les séquences : le chien poursuit le chat et : le chat poursuit le chien risqueraient d'être insuffisamment distinguées par une opération symétrique.

8.5. Query, Key et Value

À partir de la matrice d'entrée (X), le Transformer produit trois projections :

où :

- (Q) signifie Query ;
- (K) signifie Key ;
- (V) signifie Value.

Ces mots peuvent induire en erreur s'ils sont interprétés trop littéralement.

Ils désignent trois rôles mathématiques. Query La requête représente ce qu'une position cherche relativement aux autres positions. Key La clé représente la manière dont une position peut être comparée à une requête. Value La valeur représente l'information qui sera agrégée si cette position reçoit du poids. Le mécanisme sépare donc :

- le calcul de compatibilité ;
- le contenu transporté.

8.6. Le score d'attention

Pour une position (i) et une position (j) :

Le produit scalaire mesure une compatibilité entre la requête de (i) et la clé de (j).

La division par (\sqrt{d_k}) stabilise l'échelle.

Les scores sont ensuite transformés par softmax :

Ainsi :

Chaque position reçoit une distribution de poids sur les autres positions. La sortie est :

La position (i) est donc reconstruite à partir d'une combinaison contextuelle des valeurs.

8.7. Ce que réalise réellement l'attention

L'attention produit une agrégation dépendante du contexte. Elle ne dit pas directement : ce mot est le sujet grammatical. Elle calcule : compte tenu des représentations actuelles, quelles autres positions doivent contribuer à la mise à jour de cette position ? La relation est apprise et dynamique.

Elle dépend :

- des paramètres appris ;
- de la couche ;
- de la tête d'attention ;
- de la séquence ;
- de la position ;
- de l'état courant.

Ainsi, la matrice d'attention change selon l'entrée.

8.8. Attention et transformation relationnelle

L'attention peut être décrite comme une transformation relationnelle distribuée.

Le modèle produit d'abord une structure de pondération :

puis l'utilise pour transformer les valeurs. Comme dans une involution, une transformation produit un opérateur dépendant de l'entrée. Mais ici, l'opérateur est une matrice de relations entre positions.

8.9. Attention n'est pas explication

Une matrice d'attention peut être visualisée. Cela ne signifie pas qu'elle constitue à elle seule une explication complète de la décision du modèle. Plusieurs raisons l'interdisent.

- Il existe plusieurs têtes.
- Il existe plusieurs couches.
- Les sorties sont mélangées.
- Les connexions résiduelles conservent d'autres informations.
- Les couches MLP transforment les représentations.
- Le décodage final dépend de toute la chaîne.
- Des matrices d'attention différentes peuvent parfois produire des

sorties proches.

L'attention fournit une trace partielle du calcul. Elle ne résume pas l'intégralité du mécanisme causal.

8.10. Multi-head attention

Un Transformer utilise généralement plusieurs têtes d'attention. Pour chaque tête (h) :

puis :

Les têtes sont concaténées :

Chaque tête peut apprendre des relations différentes. Certaines peuvent devenir sensibles à :

- la proximité locale ;
- les dépendances syntaxiques ;
- les reprises ;
- la structure de phrase ;
- les motifs de format ;
- les positions particulières.

Mais cette spécialisation n'est ni garantie ni parfaitement stable.

8.11. Les connexions résiduelles

La sortie d'un bloc n'écrase pas entièrement son entrée. On utilise une connexion résiduelle :

Puis, après normalisation et passage dans un réseau feed-forward :

Les résidus facilitent l'optimisation et conservent une voie directe pour l'information. Dans une lecture informationnelle, ils empêchent chaque transformation de remplacer totalement l'état précédent. Le nouvel état devient :

é

Cette structure ressemble à une mise à jour incrémentale.

8.12. Le réseau feed-forward

Après l'attention, chaque position traverse généralement un réseau non linéaire. Une forme simplifiée :

L'attention mélange l'information entre positions. Le MLP transforme ensuite chaque représentation. Ainsi :

Le bloc Transformer alterne donc :

- interaction ;
- transformation locale ;
- conservation résiduelle ;
- normalisation.

8.13. La profondeur

Une couche d'attention ne suffit pas à construire toutes les relations utiles. Les couches successives produisent une hiérarchie contextuelle :

Chaque couche travaille sur les représentations produites par la précédente. La relation calculée à la couche (l) dépend donc de l'histoire du calcul.

La structure relationnelle est dynamique et profonde.

8.14. Modèle causal

Dans un modèle génératif causal, une position ne peut généralement regarder que les positions précédentes. On utilise un masque :

Le score devient :

Ainsi, pour prédire le token suivant, le modèle n'accède pas au futur. La séquence est générée pas à pas.

8.15. Prédiction du token suivant

L'objectif d'apprentissage fondamental d'un LLM causal peut être écrit :

Le modèle apprend à estimer :

Il ne reçoit pas directement une règle universelle du langage. Il apprend des régularités statistiques permettant de prédire la continuation. Cette tâche simple en apparence contraint le modèle à apprendre de nombreuses structures :

- grammaire ;
- style ;
- connaissances ;
- relations conceptuelles ;
- motifs de raisonnement ;
- formats ;
- régularités sociales.

La capacité émergente provient de la richesse de l'objectif et des données.

8.16. Générer n'est pas retrouver

Un moteur de recherche sélectionne des documents existants. Un LLM produit une distribution sur des tokens.

Puis une stratégie de décodage choisit un token.

La sortie est ensuite réinjectée :

La génération est donc récursive. Le modèle construit progressivement un chemin.

8.17. Le décodage comme décision

La distribution produite par le modèle n'est pas encore la sortie finale. Il faut une règle de décodage. Maximum

Température

Top-k On conserve les (k) tokens les plus probables. Top-p On conserve le plus petit ensemble dont la probabilité cumulée dépasse (p). Ainsi :

è é é

Le même modèle peut produire des comportements différents selon le décodage.

8.18. Représentation, comparaison et décision dans le Transformer

Nous pouvons décomposer un LLM ainsi :

Chaque étape possède une fonction différente.

- (\tau) segmente ;
- (E) représente ;
- (\mathcal{T}_\theta) contextualise ;
- (W_{\text{vocab}}) projette ;
- softmax normalise ;
- (\Delta) décide.

Cette séparation évite d'attribuer toutes les propriétés au seul mot « Transformer ».

8.19. Mémoire du contexte

Un LLM standard utilise le contexte fourni dans sa fenêtre. Il ne possède pas nécessairement une mémoire persistante au sens d'une base relationnelle durable. Son état dépend de :

- ses paramètres ;
- les tokens présents ;
- les éventuels outils externes ;
- les caches d'attention ;
- les instructions.

Une information absente du contexte peut ne pas être activée de manière fiable. Une information présente peut être oubliée au cours d'une longue séquence ou insuffisamment pondérée. Il faut donc distinguer :

é é
é
é
é

Ces quatre objets ne sont pas équivalents.

8.20. Mémoire paramétrique

Les paramètres du modèle condensent des régularités acquises durant l'apprentissage. Cette mémoire est :

- distribuée ;
- difficile à localiser ;
- difficile à modifier ponctuellement ;
- fortement compressée ;
- dépendante de l'entraînement.

Elle ne ressemble pas à une base de faits explicite. Modifier un fait isolé peut nécessiter une intervention complexe.

8.21. Mémoire de contexte

La fenêtre de contexte contient les tokens disponibles durant l'inférence. Elle est :

- temporaire ;
- explicite ;
- limitée ;
- sensible à l'ordre ;
- recalculée ou mise en cache.

Le modèle peut utiliser cette information sans l'intégrer durablement à ses paramètres.

8.22. Mémoire externe

Une architecture RAG peut interroger une base externe.

Le LLM devient alors un composant d'un système plus large. La qualité dépend :

- de l'indexation ;
- de la requête ;
- de la récupération ;
- du classement ;
- de la contextualisation ;
- de la génération.

Un échec ne doit pas être attribué automatiquement au modèle.

8.23. La représentation contextuelle

Le même token peut recevoir des états différents.

Ainsi :

Le Transformer construit donc une qualification contextuelle implicite. Mais cette qualification est distribuée dans le calcul. Elle n'est pas nécessairement produite comme un objet externe nommé :

L'ÉTR peut extraire ou reconstruire une telle qualification, mais elle n'est pas donnée par défaut.

8.24. Hallucination

Un LLM produit le token le plus compatible avec son état et sa politique de décodage. Il ne vérifie pas automatiquement chaque affirmation dans une source externe. Une séquence peut donc être :

- linguistiquement cohérente ;
- statistiquement plausible ;
- factuellement fausse.

Cette propriété découle de l'objectif.

é é é

L'hallucination n'est pas une anomalie inexplicable. Elle est un risque structurel lorsque la génération n'est pas contrainte par une vérification externe.

8.25. Confiance linguistique et confiance épistémique

Un modèle peut produire une phrase avec une forte probabilité locale. Cela ne signifie pas que le contenu est vrai. Il faut distinguer :

la cohérence linguistique, de :

é

la confiance épistémique. Une formulation fluide peut avoir :

é é

et :

é

L'ÉTR doit conserver cette séparation.

8.26. LLM et explication

Un LLM peut expliquer une réponse. Mais cette explication est elle-même une génération. Elle peut :

- refléter partiellement le raisonnement ;
- reconstruire une justification plausible ;
- omettre certaines causes ;
- rationaliser après coup ;
- être correcte sans être causalement exacte.

Il faut donc distinguer :

et :

L'ÉTR s'intéresse principalement à la seconde lorsqu'elle est disponible, et à la provenance de la première lorsqu'elle ne l'est pas.

8.27. LLM comme producteur de qualifications

Dans une architecture ÉTR, un LLM peut remplir plusieurs rôles. Qualification sémantique

Proposition de relations

Extraction structurée

é

Génération de candidats

Explication

Dans tous les cas, la sortie doit être accompagnée de :

- provenance ;
- version du modèle ;
- prompt ;
- paramètres ;
- incertitude ;
- validation éventuelle.

8.28. LLM et décision

Le LLM peut proposer une décision. Il ne doit pas nécessairement être l'autorité finale. Une architecture contrôlée peut écrire :

é

puis :

é é

Le routeur peut :

- accepter ;
- refuser ;
- demander une vérification ;
- consulter une source ;
- solliciter un humain ;
- différer.

Ainsi, la génération et la gouvernance sont séparées.

8.29. Comparaison Transformer / ÉTR

Dimension	Transformer	ÉTR
Objet central	représentation contextuelle	transformation déclarée
Relation	poids internes appris	objet typé et traçable
Mémoire	paramètres, contexte, outils	états, chemins, référentiels, provenances
Objectif	prédiction ou transformation de séquence	description, comparaison, routage, mise à jour
Interprétabilité	partielle et indirecte	exigence architecturale
Qualification	distribuée dans le calcul	transformation distincte
Décision	décodage ou tête de	routeur explicite

sortie

```
Perte                    généralement implicite déclarée par contrat
Réversibilité            non garantie           propriété déclarée
Compatibilité            modèle exécutable      cadre intégrateur
```

8.30. Complémentarité

Le Transformer peut produire des lectures sémantiques extrêmement riches. L'ÉTR peut leur fournir :

- un statut ;
- un type ;
- une provenance ;
- un référentiel ;
- une limite ;
- un chemin ;
- un contrat de mise à jour.

L'architecture combinée peut être :

é É é

Le Transformer apporte la puissance d'inférence. L'ÉTR apporte la discipline de transformation.

8.31. Exemple : question documentaire

Un utilisateur demande : Quelles sont les causes probables de cette défaillance industrielle ? Le système peut suivre ce pipeline.

1. Lecture de la question

2. Extraction des contraintes

- type de machine ;
- symptôme ;
- date ;
- environnement ;
- historique.

3. Recherche documentaire

4. Lecture des documents par le LLM

5. Construction d’un profil relationnel

6. Vérification des contradictions

7. Décision

é

8. Mise à jour

Le LLM ne constitue qu’un composant. Le système complet produit une réponse plus traçable.

Conclusion du chapitre

Le Transformer a déplacé le centre du calcul. Le CNN applique des filtres locaux partagés. L’involution génère des filtres locaux dépendants du contexte. Le Transformer construit des

relations globales entre positions et utilise ces relations pour reconfigurer les représentations. La progression peut être résumée ainsi :

Mais, dans tous les cas, la question de la déclaration reste ouverte. Le système calcule des transformations.

L'ÉTR demande : lesquelles, dans quel référentiel, avec quelles pertes, pour quelle décision et avec quelle mémoire ? Le prochain chapitre abordera les graphes de connaissances et les réseaux neuronaux sur graphes. Ils occupent une position particulière. Contrairement aux Transformers, ils représentent souvent les relations de manière explicite. Mais représenter explicitement une relation ne signifie pas encore représenter explicitement sa transformation.

Chapitre 9 — Les graphes de connaissances : représenter explicitement des relations

Les architectures étudiées jusqu'ici représentent principalement l'information sous forme de séquences, de tenseurs ou de vecteurs. Les graphes adoptent une autre logique. Ils partent d'une idée simple : certains objets ne sont correctement décrits qu'à travers les relations qu'ils entretiennent avec d'autres objets. Un graphe contient généralement :

- des nœuds ;
- des arêtes ;
- éventuellement des types ;
- éventuellement des propriétés ;
- éventuellement une orientation.

On peut écrire :

où :

- (V) est l'ensemble des nœuds ;
- (E\subseteq V\times V) est l'ensemble des relations.

Dans un graphe de connaissances, les arêtes sont souvent typées.

signifie : l'objet (u) entretient la relation (r) avec l'objet (v). Par exemple :

[redacted]

La relation devient ici visible. Elle n’est plus seulement implicite dans une distance ou une activation. C’est une évolution conceptuelle majeure.

9.1. Le triplet comme unité minimale

Le triplet relationnel peut être écrit :

où :

- (h) est la tête ;
- (r) est la relation ;
- (t) est la queue.

Cette structure encode une proposition élémentaire.

Par exemple :

ou :

Le graphe permet donc une représentation explicite des dépendances.

9.2. Relation explicite ne signifie pas relation suffisante

Le graphe nomme la relation. Mais il ne décrit pas nécessairement :

- comment cette relation a été obtenue ;
- dans quel contexte elle est valide ;
- quelle transformation l'a produite ;
- si elle est réversible ;
- si elle est certaine ;
- si elle dépend du temps ;
- si elle est contestée ;
- si elle a remplacé une relation antérieure.

Ainsi :

est une assertion relationnelle. Ce n'est pas encore un contrat complet de transformation. Pour obtenir une structure plus riche, il faudrait écrire :

où :

- (R) est le référentiel ;
- (\Pi) la provenance ;
- (U) l'incertitude ;
- (T) la temporalité ;
- (\mathcal{C}) le contrat.

Le graphe de connaissances fournit donc une base naturellement compatible avec l'ÉTR, mais il doit être enrichi pour satisfaire les exigences complètes du système.

9.3. Graphe homogène et graphe hétérogène

Dans un graphe homogène, les nœuds et relations appartiennent à peu de types distincts. Dans un graphe hétérogène, plusieurs types coexistent. Par exemple :

et :

[redacted]

Un graphe hétérogène est particulièrement adapté aux environnements complexes. Mais il exige un typage rigoureux. Une relation :

ne peut pas être appliquée arbitrairement entre tous les nœuds. Il faut préciser :

Cette logique rejoint directement la notion de signature dans l'ÉTR.

9.4. Ontologie et schéma

Un graphe de connaissances s'appuie souvent sur une ontologie. L'ontologie définit :

- les classes d'objets ;
- les types de relations ;
- les contraintes ;
- les héritages ;
- les propriétés.

Par exemple :

é

[redacted]

L'ontologie structure ce qui peut exister dans le graphe. Elle ne doit pas être confondue avec le graphe lui-même.

L'ontologie définit le langage. Le graphe contient les occurrences concrètes.

9.5. Le raisonnement symbolique

Un graphe peut permettre des inférences. Supposons :

[redacted]

Avec une règle :

[redacted]

on obtient :

[redacted]

Cette inférence est explicite. La règle est déclarée. Le chemin est identifiable. C'est l'une des grandes forces des systèmes symboliques.

9.6. Composition de relations

Soient deux relations :

Leur composition est :

Cette écriture formalise un chemin relationnel.

Mais, là encore, la composition relationnelle ne décrit pas nécessairement une transformation de l'état (a). Elle établit une relation dérivée entre (a) et (c). La distinction est essentielle :

é é

9.7. Propriété transitive et fausse transitivité

Certaines relations sont transitives.

Par exemple :

Mais de nombreuses relations ne le sont pas.

et :

n'implique pas :

De même :

à

et :

à

n'impliquent pas nécessairement une forte similarité entre (A) et (C). Le système doit donc déclarer les propriétés de chaque relation :

- réflexive ;
- symétrique ;
- antisymétrique ;
- transitive ;
- fonctionnelle ;
- inversible ;
- temporelle ;
- probabiliste.

Sans cela, le graphe peut produire des inférences illégitimes.

9.8. La temporalité

Une relation peut être vraie à un instant et fausse à un autre.

Par exemple :

peut ne plus être vrai en 2026. Il faut donc distinguer :

de :

ou :

Un graphe sans temporalité peut conserver des relations devenues obsolètes comme si elles étaient encore valides. L'ÉTR traite cette question par la provenance et le versionnement des états.

9.9. Contradictions

Un graphe réel peut contenir :

et :

ou plusieurs assertions incompatibles provenant de sources différentes. Il faut alors éviter de supprimer prématurément la contradiction. On peut conserver :

La contradiction devient un objet du système. Elle peut être :

- résolue ;
- suspendue ;
- contextualisée ;
- versionnée ;
- transmise à un humain.

Une mémoire scientifique sérieuse ne doit pas transformer automatiquement l'incertitude en unicité.

9.10. Les graphes probabilistes

Certaines relations sont pondérées.

Ce poids peut représenter :

- une probabilité ;
- une confiance ;
- une fréquence ;
- une intensité ;
- une proximité.

Mais une nouvelle fois, la signification du poids doit être déclarée.

n’a aucun sens sans type. Il faut écrire :

Le graphe pondéré ne résout donc pas à lui seul la question de l’interprétation.

9.11. Les embeddings de graphes

Un graphe peut être projeté dans un espace vectoriel. Chaque nœud reçoit :

Chaque relation peut aussi être représentée. Par exemple, dans certaines méthodes :

Le graphe symbolique devient alors compatible avec la recherche vectorielle. Cette hybridation permet :

- la prédiction de liens ;
- la détection de similarités ;
- la complétion ;
- le clustering ;
- la recommandation.

Mais elle réintroduit les mêmes problèmes que les embeddings classiques :

- perte d’explicitation ;
- compression ;
- difficulté d’interprétation ;
- dépendance au modèle.

9.12. Graphe de connaissances et ÉTR

La compatibilité est forte. Le graphe fournit :

- les objets ;
- les relations ;
- les chemins ;
- les types ;
- les voisinages.

L'ÉTR ajoute :

- les transformations ;
- les contrats ;
- les pertes ;
- les états ;
- la provenance ;
- la qualification ;
- le routage ;
- la mémoire d'évolution.

On pourrait écrire :

mais l'ÉTR demande également :

é

9.13. Exemple : graphe scientifique

Un système de recherche scientifique peut contenir :

- articles ;
- auteurs ;
- méthodes ;
- hypothèses ;
- résultats ;
- données ;
- contradictions.

Relations :

é

è

é é

L'ÉTR peut alors conserver non seulement les relations, mais les transformations scientifiques :

è é

Le graphe décrit l'état du savoir. L'ÉTR décrit son évolution.

Chapitre 10 — Les réseaux neuronaux sur graphes : apprendre par propagation relationnelle

Les graphes de connaissances rendent les relations explicites. Les réseaux neuronaux sur graphes, ou GNN, ajoutent une capacité d'apprentissage. Ils cherchent à produire des représentations en utilisant la structure du graphe. Le principe général est : un nœud peut être compris à partir de ses propriétés et de celles de ses voisins. Un nœud ne dépend donc pas uniquement de lui-même. Il dépend de son environnement relationnel.

10.1. État initial des nœuds

Chaque nœud (v) possède une représentation initiale :

Cette représentation peut contenir :

- des attributs ;
- un type ;
- un embedding ;
- des mesures ;
- une identité encodée.

Le GNN produit ensuite des états successifs :

10.2. Message passing

Une forme générale de propagation est :

puis :

où :

- (\mathcal{N}(v)) est le voisinage de (v) ;
- (M_l) construit les messages ;
- (\operatorname{AGG}) les agrège ;
- (U_l) met à jour l'état du nœud.

Cette structure sépare :

- production de messages ;
- agrégation ;
- mise à jour.

Elle est donc naturellement lisible dans le langage de l'ÉTR.

10.3. Le message comme transformation locale

Pour chaque arête :

un message est calculé :

Ce message représente l'influence de (u) sur (v). Il ne constitue pas nécessairement une relation symbolique. Il s'agit d'une transformation numérique apprise. Le GNN convertit donc une relation explicite en contribution latente.

10.4. Agrégation

Les messages entrants sont agrégés. Exemples :

Cette opération doit souvent être invariante à la permutation des voisins. Si l'ordre des voisins n'a pas de sens :

pour toute permutation (\pi).

Cette invariance est adaptée aux ensembles. Mais elle peut supprimer l'ordre lorsqu'il est pertinent.

10.5. Mise à jour du nœud

Après agrégation :

Le nouvel état combine :

- l'identité antérieure ;
- l'information du voisinage ;
- les paramètres appris.

Un nœud devient progressivement une représentation de son contexte local. Après (L) couches, il dépend approximativement de son voisinage à (L) sauts.

10.6. Profondeur et rayon relationnel

Une couche lit les voisins directs. Deux couches lisent les voisins des voisins.

La profondeur élargit donc le rayon relationnel. Mais elle peut produire deux difficultés. Sur-lissage Les représentations deviennent trop similaires.

pour de nombreux nœuds. Sur-écrasement Trop d'informations doivent être condensées dans une représentation de dimension limitée. Le système perd alors certaines distinctions.

10.7. GCN simplifié

Une couche de convolution sur graphe peut être écrite :

où :

- (\widetilde{A}=A+I) ajoute les boucles propres ;
- (\widetilde{D}) est la matrice de degrés ;
- (H^{(l)}) contient les états ;
- (W^{(l)}) contient les paramètres ;
- (\sigma) est une activation.

Cette formule montre que les états sont diffusés selon la structure du graphe.

10.8. Attention sur graphe

Un GNN peut utiliser l'attention. Pour une arête ((u,v)) :

Puis :

Le réseau apprend quels voisins comptent davantage. On retrouve alors un principe proche du Transformer, mais contraint par le graphe. Le Transformer peut connecter de nombreuses positions. Le GNN respecte généralement la structure relationnelle disponible.

10.9. Graphe donné et graphe appris

Un GNN suppose souvent qu'un graphe existe déjà. Mais ce graphe peut être :

- construit par des experts ;
- extrait automatiquement ;
- produit par proximité ;
- appris ;
- dynamique.

La qualité du GNN dépend alors de la qualité du graphe.

é

Le modèle peut apprendre à pondérer les relations, mais il ne corrige pas toujours une ontologie mal définie.

10.10. Relations typées

Dans un graphe hétérogène, les relations peuvent avoir des transformations distinctes.

Chaque type (r) possède ses paramètres. Par exemple :

Cette séparation est essentielle. Toutes les arêtes ne produisent pas le même type d'influence.

10.11. Sens de propagation

Une relation orientée ne doit pas être traitée comme symétrique.

n'implique pas :

Dans certains systèmes, on introduit une relation inverse :

Par exemple :

et :

[redacted]

Cette inversion doit être déclarée. Elle n'est pas toujours une simple symétrie.

10.12. Le GNN comme système de qualification relationnelle

Un GNN transforme l'état d'un nœud selon son environnement. On peut écrire :

La qualification dépend du voisinage. Cette lecture est utile. Mais elle ne doit pas conduire à assimiler automatiquement le GNN à l'ÉTR. Le GNN apprend une fonction. L'ÉTR décrit le contrat de cette fonction, ses effets et son inscription dans une mémoire.

10.13. Traçabilité des messages

Un GNN standard peut agréger plusieurs messages :

Après agrégation, les contributions individuelles peuvent devenir difficiles à isoler. Si l'on souhaite une traçabilité forte, il faut conserver :

ainsi que :

- leurs poids ;
- leurs types ;
- leur provenance ;
- leur ordre éventuel ;
- leur effet sur la mise à jour.

Sans cela, le GNN produit une représentation efficace, mais le chemin relationnel est compressé.

10.14. Comparaison avec le Transformer

Le Transformer construit un graphe dense implicite entre positions. Le GNN utilise un graphe explicite, souvent plus parcimonieux. On peut résumer :

é

Le premier part d'une séquence. Le second part d'un graphe. Les deux utilisent des transformations contextuelles.

10.15. Comparaison avec le CNN

Le CNN exploite un voisinage régulier. Le GNN exploite un voisinage irrégulier. Dans une image :

est défini par la grille. Dans un graphe :

est défini par les arêtes. Le GNN peut donc être compris comme une généralisation de certaines opérations locales à des structures non euclidiennes.

10.16. Limites du GNN

Les principales limites incluent :

- dépendance à la qualité du graphe ;
- sur-lissage ;
- sur-écrasement ;
- difficulté à représenter certaines structures globales ;
- coût sur les très grands graphes ;
- propagation de biais ;
- faible explicitation des transformations internes ;
- sensibilité aux relations manquantes ou erronées.

Un graphe explicite ne garantit donc pas une intelligence explicite. La propagation neuronale réintroduit une représentation latente.

10.17. GNN et ÉTR

Une architecture combinée pourrait fonctionner ainsi :

où (H_t) contient les états appris. Puis :

É

où (\mathcal{R}_t) contient :

- les relations qualifiées ;
- les transformations ;
- les incertitudes ;
- les chemins ;
- les décisions.

Le GNN permet l'inférence relationnelle latente. L'ÉTR permet la déclaration et l'organisation de cette inférence.

10.18. Exemple : réseau d'entreprise

Les nœuds sont :

- personnes ;
- équipes ;
- projets ;
- documents ;
- compétences.

Les arêtes sont :

- participe à ;
- possède ;
- dépend de ;
- consulte ;
- produit.

Un GNN peut détecter :

- les compétences proches ;
- les risques de dépendance ;
- les équipes isolées ;
- les personnes centrales ;
- les projets vulnérables.

L'ÉTR peut ensuite déclarer :

- quels chemins ont produit le diagnostic ;
- quelles relations ont été utilisées ;
- quelles données manquent ;
- quelles transformations ont modifié le profil ;
- quelle action est recommandée ;
- quelles conséquences restent incertaines.

Le GNN détecte une structure. L'ÉTR transforme cette détection en objet gouvernable.

Chapitre 11 — Les systèmes multi-agents : lorsque plusieurs unités décident et transforment

Les systèmes multi-agents introduisent un niveau supplémentaire. Jusqu'ici, nous avons surtout étudié des architectures qui reçoivent une entrée et produisent une sortie. Un système multi-agent contient plusieurs unités capables de :

- percevoir ;
- raisonner ;
- communiquer ;
- décider ;
- agir ;
- mémoriser.

Un agent peut être humain, logiciel, robotique ou hybride. On peut représenter un agent (a_i) par :

où :

- (P_i) est la perception ;
- (M_i) la mémoire ;
- (D_i) la décision ;
- (A_i) l'action ;
- (G_i) les objectifs.

Le système global devient :

avec un ensemble de relations et d'interactions.

11.1. Un agent n'est pas seulement un modèle

Un LLM peut constituer le cœur cognitif d'un agent. Mais un agent complet possède également :

- un état ;
- des outils ;
- des permissions ;
- une identité ;
- un historique ;
- un environnement ;
- une politique d'action.

Ainsi :

dans de nombreuses architectures.

Le modèle génère. L'agent agit.

11.2. Boucle perception–décision–action

Une boucle simple s'écrit :

Le système est récursif. Chaque décision modifie l'environnement qui produira les perceptions suivantes.

11.3. Interaction entre agents

Deux agents peuvent échanger des messages :

Mais le message n'est pas nécessairement compris de la même manière par les deux agents.

Ils peuvent posséder :

- des référentiels différents ;
- des objectifs différents ;
- des mémoires différentes ;
- des droits différents ;
- des modèles différents.

La communication ne garantit donc pas l'alignement.

11.4. Coordination et conflit

Les agents peuvent :

- coopérer ;
- négocier ;
- se concurrencer ;
- se contredire ;
- déléguer ;
- vérifier ;
- arbitrer.

Le système doit distinguer :

é

Ces notions ne sont pas équivalentes. Deux agents peuvent se coordonner sans partager le même objectif. Ils peuvent être d'accord sur une action pour des raisons différentes. Ils peuvent produire un consensus erroné.

11.5. Mémoire individuelle et mémoire collective

Chaque agent possède :

Le système peut également posséder une mémoire commune :

La mise à jour doit alors préciser :

- qui a écrit ;
- selon quelle autorité ;
- à partir de quelle preuve ;
- avec quel niveau de confiance ;
- si les autres agents ont validé ;
- si une contradiction subsiste.

Une mémoire collective sans gouvernance peut devenir incohérente.

11.6. Propagation des erreurs

Dans un système multi-agent, une erreur peut se propager.

Si chaque agent considère le message précédent comme une preuve, l'incertitude initiale peut être transformée en certitude apparente. Il faut donc conserver la provenance transitive.

Un agent ne doit pas être traité comme une source indépendante s'il répète une information provenant du même origine.

11.7. Autorité et permission

Tous les agents ne doivent pas posséder les mêmes capacités. On peut définir :

Un agent peut proposer une action sans pouvoir l'exécuter. Un autre peut valider sans produire. Un humain peut conserver le pouvoir final. Cette séparation est essentielle dans les systèmes sensibles.

11.8. Le routeur

Un routeur détermine quel agent doit intervenir.

Mais la sélection peut dépendre de :

- la compétence ;
- le coût ;
- la disponibilité ;
- la confiance ;
- les permissions ;
- le domaine ;
- le risque.

Le routeur constitue donc une décision relationnelle. Il compare une tâche à des profils d'agents.

11.9. ÉTR et systèmes multi-agents

L'ÉTR peut fournir un langage commun. Chaque agent produit :

é

Le système peut alors comparer :

- les transformations proposées ;
- les référentiels ;
- les contradictions ;
- les coûts ;
- les conséquences.

Le choix final ne repose plus uniquement sur un vote ou une confiance globale. Il repose sur la structure des transformations.

11.10. Consensus relationnel

Un consensus simple peut être :

é

Mais une majorité ne garantit pas la validité. L'ÉTR permet un consensus pondéré par la qualité des relations :

Le routeur peut tenir compte :

- de la provenance ;
- de la compatibilité des référentiels ;
- de la couverture des preuves ;
- de la redondance des sources ;
- des conflits non résolus.

Ainsi, trois agents répétant la même source ne valent pas nécessairement trois confirmations indépendantes.

11.11. Agent comme transformation continue

Un agent ne produit pas seulement des réponses. Il transforme un environnement.

Cette action peut être :

- réversible ;
- irréversible ;
- locale ;
- globale ;
- coûteuse ;
- risquée.

Le contrat d'action doit donc déclarer :

é é é é

Cette exigence devient cruciale pour la robotique et l'entreprise.

11.12. Exemple : équipe d'agents scientifiques

Un agent recherche les publications. Un agent extrait les méthodes. Un agent vérifie les statistiques. Un agent cherche les contradictions. Un agent synthétise. La chaîne devient :

Sans traçabilité, le résultat final peut sembler cohérent tout en reposant sur une erreur ancienne. Avec l'ÉTR, chaque passage devient un objet inspectable.

Conclusion de la deuxième partie

Les architectures étudiées répondent à des problèmes distincts. Le moteur documentaire retrouve. L'embedding rapproche. Le CNN extrait des motifs locaux. L'involution adapte l'opérateur au contexte. Le Transformer construit des représentations contextuelles globales. Le graphe explicite les relations.

Le GNN propage l'information selon ces relations. Le système multi-agent distribue perception, décision et action entre plusieurs unités. Aucune de ces architectures n'est réductible à une autre. Elles peuvent être combinées. Leur complémentarité peut être représentée ainsi :

é é é é

L'ÉTR intervient comme langage transversal. Elle ne demande pas : quel modèle doit remplacer les autres ? Elle demande : comment déclarer les transformations qui relient leurs productions ? La

partie suivante introduira donc directement le système ÉTR. Nous passerons des comparaisons externes à son architecture interne :

- objets ;
- états ;
- relations ;
- transformations ;
- référentiels ;
- Compose ;
- qualification ;
- décision ;
- mémoire ;
- validation.

Oui — il manque bien le Chapitre 12. La transition précédente annonçait directement la troisième partie, mais le numéro n'a pas été matérialisé. Il doit précisément remplir la fonction que tu avais posée dès le début : expliquer l'ensemble du système ÉTR avant d'en détailler chaque objet.

TROISIÈME PARTIE

Comprendre l’Équation des Transformations Relationnelles

Chapitre 12 — Vue d’ensemble du système ÉTR

L’Équation des Transformations Relationnelles ne désigne pas une équation unique qui résoudrait tous les problèmes relationnels. Elle désigne un système scientifique plus large, composé :

- d’une ontologie ;
- de transformations déclarées ;
- de référentiels ;
- de règles de composition ;
- de mécanismes de qualification ;
- de fonctions de comparaison ;
- d’un routeur de décision ;
- d’une mémoire évolutive ;
- de protocoles de validation.

Le mot « équation » doit donc être compris dans un sens architectural. L’ÉTR cherche à formaliser le passage :

é é

tout en conservant les conditions ayant rendu ce passage possible. Elle ne cherche pas seulement à produire une sortie. Elle cherche à décrire :

- ce qui est entré ;
- ce qui a été lu ;
- ce qui a été transformé ;
- ce qui a été conservé ;
- ce qui a été ajouté ;
- ce qui a été perdu ;
- ce qui reste incertain ;
- ce qui a conduit à la décision ;
- ce qui doit être inscrit en mémoire.

La forme générale du système peut être représentée ainsi :

é é

Cette chaîne constitue le squelette fonctionnel de l'ÉTR.

12.1. Le problème auquel répond l'ÉTR

De nombreux systèmes informatiques produisent d'excellents résultats sans rendre explicites les transformations qui les ont produits. Un Transformer peut générer une réponse cohérente. Un CNN peut reconnaître un défaut. Un moteur vectoriel peut retrouver un document proche. Un GNN peut prédire une relation entre deux nœuds. Mais, dans chacun de ces cas, plusieurs questions restent souvent séparées du calcul principal :

- Quelles informations ont été utilisées ?
- Quelle représentation a été choisie ?
- Dans quel référentiel le résultat est-il valide ?
- Quelles transformations intermédiaires ont été appliquées ?
- Une autre transformation aurait-elle produit le même résultat ?
- Quelle information a disparu ?
- La décision est-elle réversible ?
- Le résultat doit-il modifier la mémoire ?

L'ÉTR ne prétend pas que les architectures existantes sont incapables de fournir ces informations. Elle affirme seulement qu'elles ne les considèrent pas nécessairement comme leur objet scientifique central. L'ÉTR place donc la transformation déclarée au centre du système.

12.2. L'objet central : la transformation déclarée

Une transformation classique peut être écrite :

Cette écriture indique qu'un objet (x) devient un objet (y). Mais elle reste insuffisante pour un système traçable. L'ÉTR considère qu'une transformation complète doit inclure davantage d'informations :

où :

- (\operatorname{type}) décrit la famille de transformation ;
- (\operatorname{dom}) est son domaine ;
- (\operatorname{cod}) est son codomaine ;
- (\operatorname{apply}) est son mécanisme ;
- (\operatorname{pre}) contient les préconditions ;
- (\operatorname{post}) contient les postconditions ;
- (\operatorname{inv}) indique ses propriétés d'inversion ;
- (\Pi) contient la provenance ;
- (U) représente l'incertitude.

Une transformation n'est donc pas définie uniquement par son résultat. Elle est définie par son contrat.

12.3. Une transformation doit déclarer ses effets informationnels

Toute transformation doit préciser cinq catégories.

avec :

- (I_c) : informations conservées ;
- (I_t) : informations transformées ;
- (I_a) : informations ajoutées ;
- (I_s) : informations supprimées ;
- (I_{\varnothing}) : informations devenues indéterminables.

Cette règle est fondamentale. Supposons qu'une phrase soit transformée en embedding. La transformation peut conserver :

- une proximité thématique ;
- certaines régularités sémantiques ;
- certains traits contextuels.

Elle transforme :

- une séquence symbolique en vecteur.

Elle peut rendre indéterminables :

- la formulation exacte ;
- l'ordre complet ;
- l'origine de chaque dimension ;
- certaines contradictions locales.

Le vecteur reste utile. Mais son contrat devient explicite.

12.4. Les niveaux du système

L'ÉTR sépare plusieurs niveaux qui sont souvent confondus. Ontologie

Elle définit les objets qui existent dans le système. Par exemple :

- relation ;
- transformation ;
- état ;
- chemin ;
- référentiel ;
- mémoire ;
- qualification ;
- décision.

Théorie Elle définit les propriétés que ces objets peuvent satisfaire. Par exemple :

- composition ;
- invariance ;
- inversibilité ;
- clôture ;
- cohérence ;
- admissibilité.

Algèbre Elle définit les opérations formelles. Par exemple :

Implémentation Elle indique comment les objets sont réalisés en code. Par exemple :

- Python pour la recherche ;
- Rust pour la production ;
- SQL pour la mémoire ;
- JavaScript pour la visualisation.

Application Elle indique dans quel domaine le système est utilisé. Par exemple :

- LLM ;
- vision ;
- robotique ;
- réseau social ;
- entreprise ;
- recherche documentaire.

Une implémentation ne doit pas redéfinir l'ontologie. Une application ne doit pas transformer une hypothèse locale en loi générale.

12.5. L'entrée

L'entrée peut être de nature variée :

Elle peut être :

- un texte ;
- une image ;
- un son ;
- une séquence de capteurs ;
- une requête ;
- un document ;
- un événement ;
- une action utilisateur ;
- un état organisationnel.

L'ÉTR ne suppose pas que toutes les entrées possèdent la même structure. Elle exige seulement que leur type soit déclaré.

Le type détermine les lectures admissibles.

12.6. La lecture

Une lecture transforme l'entrée en un état exploitable.

Cette lecture peut être produite par :

- un tokenizer ;
- un CNN ;
- un Transformer ;
- un parseur ;
- un capteur ;
- un moteur de règles ;
- un humain ;
- une combinaison de plusieurs systèmes.

L'ÉTR ne fixe pas une méthode unique de lecture. Elle impose que la méthode soit déclarée.

où :

- (R) est le référentiel de lecture ;
- (\Pi) contient la provenance.

Le même objet peut recevoir plusieurs lectures.

Ces lectures peuvent être comparées sans être immédiatement fusionnées.

12.7. L’état relationnel

Un état relationnel n’est pas nécessairement une position géométrique unique. Il peut contenir :

- une structure ;
- des relations ;
- des dimensions ;
- un chemin ;
- une provenance ;
- une incertitude.

On peut écrire :

où :

- (S) est la structure ;
- (\mathcal{R}) contient les relations ;
- (\Gamma) est le chemin ;
- (\Pi) est la provenance ;
- (U) est l’incertitude.

L’état constitue donc une lecture compacte, mais non nécessairement réductrice à un seul vecteur.

12.8. Compose

Compose construit un état ordonné à partir d’une séquence. Soit :

Chaque unité (a_i) produit ou sélectionne une transformation :

L’état final devient :

Si les transformations ne commutent pas :

alors l'ordre est conservé par la construction elle-même. Compose sert ainsi à représenter une séquence comme un chemin et non comme une simple collection.

12.9. Pourquoi Compose ne suffit pas

Compose produit une structure ordonnée. Mais une structure ordonnée ne constitue pas encore une interprétation complète. Considérons : L'avocat est mûr. et : L'avocat présente sa défense. Le terme « avocat » possède la même identité lexicale. Sa qualification varie selon le contexte. L'ÉTR sépare donc :

de :

Cette séparation évite de confondre structure et sens.

12.10. La qualification

Une qualification ajoute une lecture contextuelle.

Elle peut être produite par :

- un LLM ;
- une règle symbolique ;
- un expert ;
- un classificateur ;
- un GNN ;
- un moteur multimodal.

La qualification peut préciser :

- une intention ;
- un domaine ;
- une polarité ;
- une fonction ;
- une entité ;
- un rôle ;
- un niveau de confiance.

Mais elle doit rester contrainte.

où (\mathcal{A}_{\kappa}) représente le domaine admissible de

qualification. L'objet fondamental n'est donc pas une qualification isolée. C'est l'espace des qualifications autorisées.

12.11. Pourquoi borner la qualification

Sans limite, une qualification pourrait produire n'importe quelle interprétation. Le système pourrait toujours expliquer un résultat en modifiant le contexte après coup. Il deviendrait alors impossible à falsifier. Une qualification scientifique doit donc avoir :

- des entrées autorisées ;
- des sorties autorisées ;
- une amplitude maximale ;
- des préconditions ;
- des cas d'abstention ;
- des frontières de validité.

La souplesse ne doit pas devenir arbitraire.

12.12. Le référentiel

Une relation n'est interprétable que dans un référentiel. On peut représenter un référentiel par :

où :

- (\mathcal{O}) contient les objets admissibles ;
- (\mathcal{F}) contient les transformations ;
- (\mathcal{M}) contient les méthodes de comparaison ;
- (\mathcal{B}) contient les frontières ;
- (\Pi_R) contient la provenance du référentiel.

Un référentiel juridique n'utilise pas nécessairement les mêmes relations qu'un référentiel artistique. Un référentiel médical ne tolère pas la même incertitude qu'un système exploratoire. Le référentiel détermine donc les règles de lecture et de décision.

12.13. Plusieurs référentiels

Un même objet peut être lu dans plusieurs référentiels.

Par exemple, un contrat peut être lu :

- juridiquement ;
- financièrement ;
- linguistiquement ;
- historiquement ;
- organisationnellement.

Ces lectures ne doivent pas être fusionnées sans déclaration. Une transformation inter-référentielle est nécessaire :

Cette transformation doit préciser les correspondances et les pertes.

12.14. La comparabilité

Avant de comparer deux états, le système doit vérifier leur comparabilité.

Cette fonction peut retourner :

é é

La comparabilité dépend :

- du type ;
- de l'unité ;
- du référentiel ;
- de la granularité ;
- de la temporalité ;
- de la provenance ;
- des transformations de médiation disponibles.

Une distance calculée entre des objets non comparables n'a pas de signification scientifique.

12.15. La relation

Lorsque les objets sont comparables, une relation peut être calculée.

Mais l'ÉTR ne suppose pas que cette relation doive toujours être un nombre unique. Elle peut être un profil :

Par exemple :

é é é

Cette pluralité évite une agrégation trop précoce.

12.16. La relation multiplicative

Dans certaines familles de l'ÉTR, la relation entre deux états positifs peut être définie par un rapport :

—

Pour des états multidimensionnels :

— —

L'identité relationnelle est alors :

Cette relation permet de décrire une transformation proportionnelle. Mais elle ne doit pas être présentée comme l'unique famille de relations possible dans l'ÉTR. Le traité doit pouvoir accueillir plusieurs algèbres selon les objets et les tâches.

12.17. Le transport

Un transport est une transformation particulière.

Il peut être associé à une relation :

dans certains régimes. Mais toutes les transformations ne sont pas des transports. Une lecture, une qualification, une substitution ou une mise à jour ne doivent pas être forcées dans le type Transport. L'ensemble englobant est :

é

Cette distinction empêche une réduction abusive du système.

12.18. La comparaison multi-signal

Une décision peut dépendre de plusieurs relations.

é é é é

Le système ne doit pas nécessairement les fusionner immédiatement. Il peut conserver ce profil jusqu'au routeur. Deux objets peuvent obtenir le même score agrégé tout en présentant des structures très différentes. L'ÉTR préserve cette différence.

12.19. Le routeur

Le routeur reçoit un ensemble de relations et produit une orientation.

où :

- (Q) est l'état ;
- (R) est le référentiel ;
- (\Omega) est la mémoire ;
- (\mathcal{C}) contient les contraintes.

Le routeur peut décider :

- d’accepter ;
- de refuser ;
- de différer ;
- d’explorer ;
- de vérifier ;
- de consulter un humain ;
- d’interroger un autre modèle ;
- de mettre à jour la mémoire.

Le routeur est distinct de la relation. Une relation n’impose pas à elle seule une action.

12.20. Exemple de routage

Supposons qu’un document possède :

é

é

é

Dans une recherche exploratoire, il peut être affiché. Dans une décision médicale, il peut être rejeté ou soumis à vérification. Le profil relationnel reste identique. La politique de décision change.

é

La décision appartient donc au référentiel d’usage.

12.21. La décision

Une décision est une sortie formelle du routeur.

Elle peut être :

- un classement ;
- une sélection ;
- une proposition ;
- une alerte ;
- une commande ;
- une abstention ;
- une demande d'information.

Une décision n'est pas nécessairement une action. Elle peut encore nécessiter une compilation :

où (A) est une action applicable au système ou au monde.

12.22. L'action

Une action modifie un environnement.

Elle doit déclarer :

- son autorité ;
- ses préconditions ;
- ses effets ;
- son coût ;
- ses risques ;
- sa réversibilité.

Dans un système sensible, le passage de la décision à l'action peut exiger une validation humaine.

12.23. La mémoire

La mémoire relationnelle ne conserve pas uniquement des résultats. Elle conserve :

- des états ;
- des transformations ;
- des chemins ;
- des référentiels ;
- des provenances ;
- des contradictions ;
- des mises à jour.

On peut écrire :

où :

- (V) contient les états ;
- (E) contient les transformations ;
- (\Gamma) contient les chemins ;
- (R) contient les référentiels ;
- (\Pi) contient les provenances.

12.24. Update

La mise à jour transforme la mémoire.

Mais Update ne signifie pas simplement « ajouter ». Elle peut :

- créer un objet ;
- modifier une relation ;
- ajouter une provenance ;
- invalider une lecture ;
- conserver une contradiction ;
- relier deux chemins ;
- créer un nouveau référentiel ;
- refuser une information.

Le contrat d'Update doit être aussi rigoureux que celui des autres transformations.

12.25. Mémoire croissante et révision

Une mémoire croissante ne signifie pas qu'aucun élément ne peut devenir obsolète. Elle signifie que l'histoire des transformations n'est pas effacée sans déclaration. Si une relation change :

le système conserve :

- l'ancienne relation ;
- la nouvelle ;
- la transformation ;
- la date ;
- la cause ;
- la provenance.

La mémoire devient versionnée.

12.26. Le chemin

Un chemin est une suite ordonnée de transformations.

Son application est :

Deux chemins peuvent produire le même résultat :

sans être équivalents.

Ils peuvent différer par :

- le coût ;
- la perte ;
- la provenance ;
- la réversibilité ;
- le niveau de confiance.

L'ÉTR compare donc également les chemins.

12.27. Les cycles

Un cycle revient vers un état de départ.

avec une comparaison entre :

et :

Le transport total peut être :

Si :

le cycle revient à l'identité dans le cadre déclaré. Si :

un résidu apparaît. Ce résidu peut indiquer :

- une perte ;
- une dépendance au chemin ;
- une incohérence ;
- une transformation non inversible ;
- une qualification contextuelle ;
- une erreur de mesure.

Il constitue un objet d'analyse, non automatiquement une faute.

12.28. L'incertitude

L'incertitude ne doit pas être réduite à une unique probabilité. Elle peut provenir :

- des données ;
- du modèle ;
- du référentiel ;
- de la qualification ;
- de la comparaison ;
- de la décision.

On peut écrire :

é è é

Cette structure évite de produire une confiance globale artificiellement précise.

12.29. L'abstention

Un système relationnel rigoureux doit pouvoir ne pas conclure.

L'abstention peut être déclenchée lorsque :

- les objets ne sont pas comparables ;
- le référentiel est absent ;
- la qualification dépasse le domaine admissible ;
- la provenance est insuffisante ;
- les contradictions ne sont pas résolues ;
- le risque dépasse un seuil.

L'abstention constitue une sortie valide. Elle n'est pas un échec d'exécution.

12.30. Les statuts scientifiques

Chaque objet ou affirmation doit recevoir un statut.

- Définition : objet fixé par le traité ;
- Axiome : principe fondateur ;
- Proposition : résultat démontré ;
- Théorème : résultat général démontré ;
- Corollaire : conséquence directe ;
- Observation : résultat expérimental ;
- Hypothèse : proposition à tester ;
- Exemple : illustration ;
- Remarque : commentaire scientifique.

Le registre historique peut également conserver :

- ([D]) défini ;
- ([C]) construit ;
- ([O]) observé ;
- ([I]) choix d'architecture ;
- ([?]) ouvert.

Cette discipline empêche de confondre définition, preuve et intuition.

12.31. L’équation générale du système

Une forme générale peut être écrite :

Le système reçoit :

- une entrée (x_t) ;
- une mémoire (\Omega_t) ;
- un référentiel (R) ;
- une provenance (\Pi).

Il produit :

- une décision (D_t) ;
- une mémoire mise à jour (\Omega_{t+1}).

Une forme développée devient :

Cette écriture présente l’ensemble du système sans imposer une implémentation unique.

12.32. Une phrase complète comme exemple

Considérons : Le robot ralentit parce qu’un obstacle est détecté. Entrée

Lecture

Compose Le système conserve l’ordre :

é é

Qualification Le LLM ou le parseur identifie :

- agent : robot ;
- action : ralentir ;
- cause : obstacle détecté ;
- relation : causalité.

Comparaison Le système compare cet état aux scénarios connus :

é é

Routage Le routeur détermine :

- comportement normal ;
- anomalie ;
- alerte ;
- besoin de vérification.

Décision

Update La mémoire conserve :

- l'événement ;
- le capteur ;
- la décision ;
- la durée ;
- la provenance.

La phrase n'est donc pas seulement comprise ou classée. Elle devient un chemin de transformation traçable.

12.33. Comparaison globale avec un LLM

Un LLM peut effectuer plusieurs opérations de cette chaîne implicitement. Il peut :

- lire ;
- qualifier ;
- comparer ;
- proposer une décision ;
- générer une explication.

Mais ces fonctions restent souvent fusionnées dans une même génération. L'ÉTR les sépare :

É è é

Le LLM peut donc être intégré sans devenir l'ensemble du système.

12.34. Comparaison globale avec un CNN

Un CNN produit une lecture perceptive.

L'ÉTR prend ensuite en charge :

- le typage ;
- la comparaison ;
- la contextualisation ;
- la décision ;
- la mémoire.

Le CNN reconnaît. L'ÉTR inscrit la reconnaissance dans une chaîne gouvernée.

12.35. Comparaison globale avec un GNN

Un GNN produit une représentation relationnelle apprise à partir d'un graphe. L'ÉTR conserve :

- les relations utilisées ;
- les messages ;
- les transformations ;
- les versions ;
- les décisions résultantes.

Le GNN propage. L'ÉTR déclare la propagation et ses effets.

12.36. Ce que l'ÉTR ne prétend pas faire

L'ÉTR ne garantit pas :

- qu'une qualification soit vraie ;
- qu'un référentiel soit complet ;
- qu'une relation soit causalement valide ;
- qu'un modèle soit exempt de biais ;
- qu'une décision soit toujours correcte ;
- qu'un chemin soit économiquement optimal ;
- qu'un système complexe devienne entièrement explicable.

Elle garantit seulement un cadre dans lequel ces questions peuvent être posées explicitement.

12.37. Ce que l'ÉTR cherche à rendre possible

L'ÉTR cherche à rendre possible :

- la séparation entre structure et sens ;
- la déclaration des transformations ;
- la conservation des chemins ;
- le contrôle des qualifications ;
- la comparaison multi-référentielle ;
- l'audit des pertes ;
- la distinction entre modèle et décision ;
- la mémoire versionnée ;
- l'abstention ;
- la falsification localisée.

Sa puissance ne repose donc pas sur une promesse d'infaillibilité. Elle repose sur une augmentation de la lisibilité scientifique.

Conclusion du chapitre

L'Équation des Transformations Relationnelles est un système destiné à organiser le passage entre information, relation, décision et mémoire. Son architecture générale est :

Chaque flèche est une transformation distincte. Chaque transformation possède un contrat. Chaque contrat doit déclarer ses effets. Chaque décision doit être rattachée à un référentiel. Chaque mise à jour doit préserver sa provenance. Cette vue d'ensemble constitue la carte du système.

LES CHAPITRES SUIVANTS POURRONT MAINTENANT ISOLER CHACUN DE SES composants :

- l'ontologie ;
- les états ;
- les transformations ;
- Compose ;
- la qualification ;
- les référentiels ;
- la comparaison ;
- le routeur ;
- la mémoire ;
- la validation.

Chapitre 13 — Ontologie minimale de l'ÉTR

L'ontologie définit les objets que le système reconnaît comme distincts. Elle ne décrit pas encore leur implémentation ni leur efficacité. Elle fixe leur identité, leur rôle et leurs relations de type. L'ÉTR repose sur une distinction stricte entre :

É é

Confondre ces catégories produit des erreurs de conception.

13.1. Objet

Un objet est une entité admissible dans un référentiel.

Il peut s'agir :

- d'un mot ;
- d'une image ;
- d'un document ;
- d'un utilisateur ;
- d'un capteur ;
- d'un événement ;
- d'une entreprise ;
- d'une hypothèse.

L'objet n'est pas nécessairement directement calculable. Il doit d'abord être lu.

13.2. Lecture

Une lecture transforme un objet en représentation exploitable.

La lecture dépend du référentiel et de la méthode. Une même entrée peut donc produire plusieurs états :

Exemple :

é

13.3. État relationnel

Un état relationnel est une représentation typée d'un objet ou d'un chemin.

où :

- (S) est la structure lisible ;
- (\Gamma) est le chemin ayant produit l'état ;
- (\Pi) est la provenance ;
- (U) est l'incertitude.

Un état n’est pas l’objet initial.

Il est une lecture de cet objet.

13.4. Relation

Une relation caractérise un lien ou un rapport entre deux états comparables.

La relation peut être :

- numérique ;
- vectorielle ;
- symbolique ;
- logique ;
- temporelle ;
- composite.

Exemple :

é é é

Une relation ne transforme pas nécessairement un état. Elle peut seulement le décrire relativement à un autre.

13.5. Transformation

Une transformation produit un nouvel état ou un nouvel objet.

Elle doit déclarer :

Une transformation est valide seulement si :

et si ses préconditions sont satisfaites.

13.6. Familles de transformations

L'ensemble des transformations est noté :

Il contient notamment :

avec :

- (\Phi) : lecture ;
- (T) : transport ;
- (\Lambda) : qualification ;
- (S) : substitution ;
- (D) : décision ;
- (\operatorname{Update}) : mise à jour.

Le transport n'est donc qu'un sous-type de transformation.

13.7. Transport

Un transport déplace un état vers un autre état compatible.

Dans un régime multiplicatif :

et :

Cette forme n'est valide que si l'opération (\odot) et les types sont

déclarés.

13.8. Qualification

Une qualification modifie la lecture d'un état sans redéfinir son identité structurelle.

Exemple :

La qualification doit appartenir à un domaine admissible :

Elle ne peut donc pas produire arbitrairement toute interprétation.

13.9. Chemin

Un chemin est une suite ordonnée de transformations.

Son application est :

L'ordre appartient au chemin.

dans le cas général. Le chemin constitue une information distincte de son résultat.

13.10. Référentiel

Un référentiel fixe les conditions de lecture, de transformation et de comparaison.

où :

- (\mathcal{O}) : objets admissibles ;
- (\mathcal{Q}) : états admissibles ;
- (\mathcal{F}) : transformations autorisées ;
- (\mathcal{L}) : relations calculables ;
- (\mathcal{B}) : frontières ;
- (\Pi_R) : provenance.

Un résultat n'est interprétable qu'avec son référentiel.

est insuffisant. Il faut :

13.11. Comparabilité

La comparabilité est un prédicat préalable à la relation.

Elle peut prendre les valeurs :

é é

Une relation ne doit pas être calculée lorsque :

Cette règle évite les comparaisons de type incorrect.

13.12. Mémoire

La mémoire conserve les états et leur évolution.

où :

- (V) : états et objets ;
- (E) : relations et transformations ;
- (\Gamma) : chemins ;
- (R) : référentiels ;
- (\Pi) : provenances.

La mémoire ne contient donc pas seulement des résultats. Elle conserve également les conditions de leur production.

13.13. Décision

Une décision sélectionne une orientation parmi plusieurs possibilités.

Une décision peut être :

é é

La décision n’est pas une relation.

Une même relation peut produire plusieurs décisions selon le référentiel.

13.14. Action

Une action modifie un système externe ou interne.

La décision peut précéder l'action :

Cette séparation permet une validation avant exécution.

13.15. Provenance

La provenance décrit l'origine d'un objet, d'une lecture ou d'une transformation.

é

Deux états formellement identiques mais issus de provenances différentes ne sont pas nécessairement équivalents.

n'implique pas :

13.16. Incertitude

L'incertitude accompagne un résultat sans le remplacer.

é é

Elle ne constitue pas automatiquement une probabilité. Son type doit être déclaré.

13.17. Frontière

Une frontière définit la limite de validité d'un objet ou d'une opération.

Elle peut concerner :

- le domaine ;
- le temps ;
- la précision ;
- la langue ;
- le type ;
- la confiance ;
- le coût.

Hors frontière :

le système doit :

- refuser ;
- transformer vers un autre référentiel ;
- ou s'abstenir.

13.18. Invariant

Un invariant est une propriété conservée par une transformation.

Exemples possibles :

- type ;
- identité ;
- ordre relatif ;
- provenance ;
- norme ;
- orientation relationnelle.

Chaque transformation doit déclarer ses invariants attendus.

13.19. Perte

Une perte est une information non conservée par une transformation.

Elle peut être :

- volontaire ;
- irréversible ;
- estimée ;
- inconnue.

La perte doit être distinguée de l'erreur. Une compression peut perdre de l'information tout en fonctionnant conformément à son contrat.

13.20. Erreur de type

Une erreur de type survient lorsqu'un objet est utilisé comme s'il appartenait à une autre catégorie. Exemples :

é

L'ontologie sert d'abord à empêcher ces confusions.

13.21. Schéma minimal

L'architecture ontologique minimale est :

avec conservation de :

13.22. Exemple concis

Entrée : Cette machine vibre anormalement. Lecture :

Qualification :

Comparaison :

Décision :

Mise à jour :

Chaque étape demeure distincte.

Conclusion du chapitre

L'ontologie minimale de l'ÉTR repose sur une règle simple :

è é é

L'objet est lu. L'état est transformé. La relation compare. Le référentiel rend la comparaison valide. Le routeur décide. L'action exécute. La mémoire conserve. La provenance permet l'audit. L'incertitude limite la conclusion. Le chapitre suivant peut maintenant traiter précisément Compose, c'est- à-dire la construction ordonnée des états et des chemins.

Chapitre 14 — Compose : construire un état ordonné

Compose est l'opération qui transforme une séquence d'unités en un état relationnel structuré. Son rôle n'est pas de produire directement le sens complet. Son rôle est plus précis :

Compose répond donc à une question centrale : Comment construire un état sans réduire une séquence à une simple collection de ses éléments ?

14.1. Entrée de Compose

Soit une séquence ordonnée :

Chaque unité (a_i) peut être :

- un caractère ;
- un token ;
- un mot ;
- une proposition ;
- un événement ;
- une observation ;
- une action ;
- un état intermédiaire.

La séquence appartient à un niveau déclaré :

où :

- (\Sigma^{(k)}) est l'alphabet du niveau (k) ;
- ((\Sigma^{(k)})^*) est l'ensemble des séquences finies construites à

ce niveau.

Exemples :

è

Compose ne doit pas mélanger plusieurs niveaux sans transformation explicite.

14.2. État initial

Compose agit à partir d'un état initial :

Cet état doit être typé. Il peut être :

- un état neutre ;
- une lecture initiale ;
- un état mémoire ;
- un contexte déjà construit ;
- un état transmis par une opération précédente.

La forme générale est :

où :

- (R) est le référentiel ;
- (\Pi) est la provenance ;
- (\Gamma_T) est le chemin des transformations appliquées.

14.3. Transformation associée à chaque unité

Chaque unité (a_i) est associée à une transformation :

La fonction (\mathcal{G}) peut produire :

- une transformation fixe ;
- une transformation dépendante de l'unité ;
- une transformation dépendante de l'état courant ;
- une transformation conditionnelle au référentiel.

L'état évolue selon :

Ainsi :

La première transformation appliquée est (A_1).

14.4. Forme stationnaire et forme contextuelle

Deux régimes doivent être distingués. Compose stationnaire Une même unité produit toujours la même transformation :

La transformation dépend uniquement de l'unité. Compose contextuel La transformation dépend aussi de l'état courant :

Dans ce cas, la même unité peut agir différemment selon le chemin précédent.

Cette distinction doit être déclarée dans toute implémentation.

14.5. Conservation de l'ordre

Compose doit différencier :

et :

Il faut donc que, dans le cas général :

Cette non-commutativité peut provenir :

- des transformations elles-mêmes ;
- de leur dépendance à l'état ;
- de la provenance positionnelle ;
- du référentiel ;
- d'une mémoire intermédiaire.

L'ordre ne doit pas être ajouté après le calcul comme une simple métadonnée si l'objectif est qu'il participe réellement à la construction de l'état.

14.6. Exemple linguistique

Considérons : Le chien poursuit le chat.

et : Le chat poursuit le chien. Les unités principales sont proches, mais leur ordre diffère. Pour la première phrase :

Pour la seconde phrase, l'ordre des transformations change.

Dans le cas général :

Compose conserve donc une différence que la simple collection de mots ne conserverait pas.

14.7. Compose ne produit pas automatiquement le sens

Il faut maintenir une séparation stricte :

puis :

Compose construit :

- l'ordre ;
- la succession ;
- les dépendances ;
- le chemin ;
- l'état structurel.

La qualification construit ou précise :

- le domaine ;
- le rôle ;
- l'intention ;
- la polarité ;
- la désambiguïsation ;
- l'interprétation contextuelle.

Ainsi :

é

Il constitue le support ordonné sur lequel une qualification peut agir.

14.8. Pourquoi cette séparation est nécessaire

Si Compose produisait directement toute interprétation, il deviendrait impossible de distinguer :

- la structure de la phrase ;
- l'interprétation du modèle ;
- le contexte utilisé ;
- la variation sémantique ;
- l'erreur de lecture.

Avec la séparation :

reste disponible même si :

Deux modèles peuvent donc qualifier différemment la même construction sans modifier son chemin structurel.

14.9. Compose hiérarchique

Compose peut être appliqué à plusieurs niveaux. Niveau caractère

produit un état de terme.

Niveau terme

produit un état de phrase. Niveau proposition

produit un état de discours ou de document. Chaque changement de niveau exige une opération déclarée :

14.10. Un résultat de Compose devient une unité

Le résultat d'un niveau peut devenir l'unité du niveau suivant.

Cela permet une construction récursive :

è

Mais cette récursivité ne doit pas effacer les niveaux inférieurs. La provenance doit conserver :

lorsque ces chemins restent nécessaires à l'audit.

14.11. Le chemin de Compose

Compose produit non seulement un état final, mais un chemin :

Ce chemin permet de reconstruire les états intermédiaires :

On obtient donc :

Cette suite permet :

- l'inspection ;
- la comparaison de chemins ;
- la détection d'une rupture ;
- l'analyse d'une perte ;
- la reprise partielle ;
- l'évaluation de la stabilité.

14.12. États intermédiaires

Les états intermédiaires ne sont pas nécessairement tous conservés en production. Deux régimes sont possibles. Conservation intégrale

Avantage :

- traçabilité maximale.

Coût :

- mémoire élevée.

Conservation sélective

Avantage :

- coût réduit.

Limite :

- reconstruction partielle.

Le protocole doit indiquer quels états sont conservés et lesquels deviennent indéterminables.

14.13. Contrat de Compose

Compose doit déclarer au minimum :

avec :

- domaine des unités ;
- type de l'état produit ;
- niveau de composition ;
- convention d'ordre ;
- dépendance ou non à l'état ;
- pertes éventuelles ;
- provenance conservée.

Préconditions :

Postcondition :

14.14. Clôture

Compose est clos sur un espace d'états si :

pour toute transformation admissible. Si une transformation produit un état hors domaine :

le système doit :

- refuser la transformation ;
- projeter l'état dans le domaine admissible ;
- changer de référentiel ;
- ou s'abstenir.

La projection ne doit jamais être implicite.

14.15. Compose partiel

Compose peut être une fonction partielle.

Cela signifie que certaines séquences ne produisent pas d'état valide. Causes possibles :

- unité inconnue ;
- type incompatible ;
- transformation inadmissible ;
- dépassement de frontière ;
- état non clos ;
- manque de contexte ;
- contradiction structurelle.

L'échec de Compose constitue un résultat exploitable.

14.16. Compose et identité

Une transformation identité peut être définie :

Elle représente l'absence de changement structurel. Pour une séquence vide :

on attend généralement :

Cette propriété doit être déclarée comme convention du système.

14.17. Compose et associativité

La composition fonctionnelle est associative :

Mais cette propriété ne signifie pas que la séquence peut être réordonnée. Associativité :

Commutativité :

Ce sont deux propriétés différentes. Compose utilise l'associativité pour regrouper les calculs, tout en conservant l'ordre des transformations.

14.18. Compose contextuel et associativité opérationnelle

Dans le cas contextuel :

les opérateurs ne sont pas connus indépendamment des états intermédiaires. La composition reste séquentiellement définie, mais elle ne peut pas toujours être pré-calculée comme un simple produit d'opérateurs fixes. On doit exécuter :

puis :

puis :

La dépendance à l'état interdit certaines optimisations. Elle augmente en revanche l'expressivité.

14.19. Modulation de Compose

Une transformation peut être modulée :

où :

- (A_i) est la transformation structurelle ;
- (\Lambda_i) est la modulation ;
- (\mathcal{M}) est la loi de combinaison.

La loi (\mathcal{M}) doit être déclarée.

Elle peut prendre plusieurs formes :

ou :

ou encore :

Ces formes ne sont pas équivalentes. L'ÉTR ne doit pas en imposer une sans justification.

14.20. Domaine admissible de modulation

La modulation doit appartenir à un domaine :

Cette contrainte empêche la qualification de remplacer totalement la structure. Le système doit vérifier :

avant l'application. Sinon :

est rejeté ou projeté selon une règle déclarée.

14.21. Invariants de Compose

Compose peut être conçu pour conserver certains invariants. Exemples :

é

Ces invariants dépendent de l'implémentation. Ils doivent être testés, non supposés.

14.22. Perte dans Compose

Compose peut perdre de l'information lorsque :

- les états intermédiaires ne sont pas conservés ;
- une opération agrège plusieurs dimensions ;
- une projection réduit la représentation ;
- une transformation n'est pas inversible ;
- une normalisation efface une magnitude ;
- une qualification remplace plusieurs hypothèses par une seule.

Le contrat global doit agréger ces pertes.

Cette union doit être comprise comme une composition de pertes, et non nécessairement comme une simple somme.

14.23. Inversion

Si chaque transformation est inversible :

alors le chemin inverse est :

et :

sous les préconditions déclarées. Mais si une seule transformation est non inversible, le chemin complet peut devenir non inversible.

14.24. Compose et mémoire

Compose peut lire et modifier une mémoire. Forme sans mise à jour :

Forme avec mémoire évolutive :

Ces deux régimes doivent être distingués. Compose ne doit pas modifier la mémoire implicitement. La mise à jour doit rester une transformation déclarée.

14.25. Compose et parallélisation

Un Compose strictement séquentiel suit :

Il possède donc une dépendance temporelle. Certaines opérations peuvent néanmoins être parallélisées si :

- elles sont indépendantes ;
- elles agissent sur des sous-séquences distinctes ;
- leur recomposition est déclarée ;
- les dépendances sont respectées.

On peut construire :

puis :

Mais (\operatorname{Merge}) devient alors une transformation distincte

avec son propre contrat.

14.26. Compose et Transformer

Un Transformer construit des représentations contextuelles à plusieurs couches. On peut le considérer comme un producteur d'états :

Mais son mécanisme n'est pas identique à Compose. Le Transformer :

- contextualise globalement ;
- agrège par attention ;
- transforme les représentations latentes.

Compose :

- expose une séquence de transformations ;
- conserve explicitement le chemin ;
- distingue structure et qualification ;
- exige un contrat par opération.

Le Transformer peut fournir les transformations ou les qualifications utilisées par Compose.

14.27. Compose et CNN

Un CNN compose également des couches :

La différence principale réside dans l'objet suivi. Le CNN conserve principalement des activations. Compose vise à conserver :

- les transformations ;
- le chemin ;
- les types ;
- la provenance ;
- les pertes.

Un CNN peut donc être encapsulé comme une transformation de lecture :

14.28. Exemple minimal d'application

Séquence :

État initial :

Transformations :

Composition :

Le résultat structurel conserve :

- l'objet concerné ;
- l'événement ;
- l'intensification ;
- leur ordre.

La qualification peut ensuite produire :

Compose n'effectue pas à lui seul ce diagnostic.

14.29. Forme algorithmique minimale

Entrées : état initial Q séquence ordonnée Γ référentiel R

Pour chaque unité a_i de Γ : vérifier le type de a_i produire ou sélectionner A_i vérifier les préconditions de A_i calculer $Q_i = A_i(Q_{i-1})$ vérifier $Q_i \in$ codomaine(A_i) enregistrer la provenance enregistrer les pertes déclarées

Retourner : état final Q_n chemin Γ_T rapport de validation Cette forme constitue le noyau opérationnel minimal.

14.30. Sortie de Compose

La sortie ne doit pas être seulement :

Elle doit idéalement contenir :

où :

- (Q_n) est l'état final ;
- (\Gamma_T) est le chemin ;
- (\Pi) est la provenance ;
- (U) est l'incertitude ;
- (L) contient les pertes ;
- (V) est le rapport de validation.

Conclusion du chapitre

Compose est l'opérateur de construction ordonnée de l'ÉTR. Sa forme générale est :

Mais sa définition complète comprend davantage que cette équation. Compose doit déclarer :

- le niveau de séquence ;
- l'état initial ;
- les transformations ;
- l'ordre ;
- le référentiel ;
- les états intermédiaires ;
- les pertes ;
- la provenance ;
- les conditions de validité.

Il ne produit pas automatiquement le sens. Il produit la structure et le chemin nécessaires à une qualification ultérieure.

è

Chapitre 15 — Les référentiels : le cadre dans lequel une relation prend son sens

Jusqu'à présent, nous avons introduit les objets, les états, les transformations et Compose. Une question demeure pourtant ouverte : Comment savoir si deux états peuvent être comparés ? La réponse est simple en apparence : Ils doivent appartenir au même cadre d'interprétation. Ce cadre est appelé référentiel. Sans référentiel, une relation ne possède pas de signification scientifique.

15.1. Pourquoi un référentiel ?

Considérons le nombre :

À lui seul, il ne signifie rien. Il peut représenter :

- une température ;
- un âge ;
- une vitesse ;
- une distance ;
- une masse ;
- une note ;
- un identifiant.

Le nombre est identique. Le sens change. Le référentiel est précisément ce qui permet de déterminer l'interprétation admissible. Autrement dit :

à é é é é é

15.2. Définition

Un référentiel est un ensemble cohérent de règles permettant de :

- définir les objets admissibles ;
- définir les états admissibles ;
- définir les transformations autorisées ;
- définir les relations calculables ;
- définir les décisions possibles.

On peut l'écrire :

où :

- (\mathcal O) est l'ensemble des objets ;
- (\mathcal Q) l'ensemble des états ;
- (\mathcal F) les transformations ;
- (\mathcal L) les relations ;
- (\mathcal D) les décisions autorisées.

Le référentiel constitue donc le cadre logique dans lequel les opérations deviennent valides.

15.3. Deux objets identiques peuvent changer de sens

Considérons le mot : avocat Dans un référentiel culinaire : fruit. Dans un référentiel juridique : profession. L'objet est identique. La lecture change. Compose n'a pas changé. Le référentiel, lui, a changé.

15.4. Les référentiels ne sont pas des modèles

Il est important de distinguer :

- un modèle ;
- un référentiel.

Un modèle calcule. Un référentiel définit les règles du calcul. Un LLM peut être remplacé. Le référentiel peut rester identique. Inversement : le même LLM peut fonctionner sous plusieurs référentiels. L'ÉTR sépare volontairement ces deux notions.

15.5. Plusieurs référentiels peuvent coexister

Un même objet peut être interprété simultanément selon plusieurs points

de vue. Une entreprise peut être décrite :

- économiquement ;
- juridiquement ;
- comptablement ;
- stratégiquement ;
- organisationnellement.

On obtient alors :

Ces états ne sont pas contradictoires. Ils correspondent à des lectures différentes.

15.6. Comparer exige un référentiel commun

Comparer deux objets suppose une règle commune. Comparer :

- 12 kilogrammes ;
- 12 kilomètres ;

n'a aucun sens. Avant toute relation, l'ÉTR vérifie donc :

Si les états ne sont pas comparables dans le référentiel considéré, la relation ne doit pas être calculée. L'absence de comparaison est ici un résultat valide.

15.7. Changer de référentiel

Il est parfois nécessaire de passer d'un référentiel à un autre. Cette opération constitue une transformation.

Elle doit déclarer :

- ce qui est conservé ;
- ce qui est transformé ;
- ce qui est perdu.

Changer de référentiel ne consiste donc pas à renommer des objets. C'est une transformation à part entière.

15.8. Le rôle des référentiels dans l'ÉTR

Les référentiels occupent une place centrale. Ils permettent :

- d'interpréter une lecture ;
- de vérifier la comparabilité ;
- d'autoriser certaines transformations ;
- de limiter les qualifications ;
- de guider les décisions.

Autrement dit, ils ne produisent pas directement les résultats. Ils définissent le cadre dans lequel ces résultats deviennent scientifiquement interprétables.

Tu as raison : le chapitre 15 doit présenter non seulement le référentiel comme cadre logique, mais aussi le Référentiel Relationnel Unitaire comme modèle opératoire, son échelle multiplicative, ses architectures de coordonnées et le mécanisme permettant de sélectionner un référentiel plutôt qu’un autre. La suite du chapitre peut être complétée ainsi.

15.9. Le Référentiel Relationnel Unitaire

Le Référentiel Relationnel Unitaire est une famille particulière de référentiels dans laquelle la relation élémentaire entre deux valeurs strictement positives est définie par leur rapport :

—

Le résultat n'exprime pas une différence additive, mais le facteur permettant de passer de (a) à (b). Exemples :

—

—

Le nombre obtenu est donc une relation de transformation :

Ainsi :

15.10. L'unité comme identité relationnelle

Dans ce modèle, l'unité (1) n'est pas nécessairement le premier point d'un axe. Elle est d'abord la relation neutre :

—

Elle signifie :

- aucune transformation ;
- conservation de l'état ;
- identité sur la dimension considérée.

Si :

alors la dimension (x) ne change pas. L'unité appartient donc à l'espace des relations avant d'être éventuellement utilisée comme coordonnée dans une représentation graphique. Cette distinction évite de confondre :

et :

é

15.11. L’échelle relationnelle

Sur un axe additif classique, les écarts :

et :

ont la même différence :

Dans une échelle relationnelle, ils ne représentent pas la même transformation :

Ainsi, l’écart graphique entre deux repères ne doit pas être interprété

automatiquement comme une unité additive constante. Chaque intervalle représente son propre facteur relationnel.

La suite générale est :

et :

Lorsque les valeurs augmentent additivement d’une unité, leur transformation relative se rapproche progressivement de l’identité.

15.12. Coordonnées multidimensionnelles

Un référentiel peut comporter plusieurs dimensions. Soient :

et :

La relation de (A) vers (B) est calculée dimension par dimension :

Pour l'exemple présenté dans les figures :

on obtient :

donc :

La lecture dimensionnelle est alors :

transformation multiplicative forte sur (x) ;

identité relationnelle sur (y) ;

transformation multiplicative modérée sur (z). Le résultat n'est donc pas une distance unique entre (A) et (B). Il constitue un profil de transformation.

15.13. Position et relation

Le modèle doit distinguer deux objets :

qui sont des états ou des coordonnées, et :

qui est la relation permettant de passer de l'un à l'autre. Dans le régime multiplicatif :

où (\odot) désigne le produit composante par composante.

Pour l'exemple :

La relation n'est donc pas la seconde position. Elle constitue la transformation entre les deux positions.

15.14. Parcours relationnel

Le passage de (A) à (B) peut être représenté comme une trajectoire paramétrée :

avec :

et une exponentiation composante par composante. Ainsi :

et :

Dans l'exemple :

À mi-parcours :

soit approximativement :

Cette trajectoire représente une évolution proportionnelle dans chacune des dimensions. Elle n'impose pas que tout phénomène réel suive cette interpolation. Elle constitue une simulation définie dans le référentiel multiplicatif considéré.

15.15. Architecture interne de coordonnées

Un référentiel peut contenir sa propre architecture de coordonnées :

Chaque coordonnée (k_i) correspond à une dimension explicitement définie dans ce référentiel. Une fonction de lecture positionne un objet :

Ainsi :

Les dimensions ne doivent pas être de simples axes anonymes. Chacune doit posséder :

- un identifiant ;
- une définition ;
- un domaine ;
- une règle de positionnement ;
- une unité ou une relation neutre ;
- des bornes ;
- une provenance ;
- un protocole de validation.

Le référentiel ne se limite donc pas à déclarer les objets comparables. Il peut aussi définir l'architecture dans laquelle ils sont situés.

15.16. Coordonnées et relations sémantiques

Dans un référentiel sémantique, les coordonnées peuvent représenter des structures relationnelles telles que :

- proximité entre concepts ;
- opposition ;
- dépendance ;
- causalité ;
- intensité ;
- temporalité ;
- appartenance ;
- compatibilité ;
- direction d'évolution.

Il faut cependant distinguer deux architectures. Coordonnées sémantiques explicites Chaque dimension possède un sens déclaré :

é é é

La relation est alors lisible dimension par dimension. Coordonnées sémantiques latentes Les coordonnées sont produites par un modèle et ne possèdent pas nécessairement une interprétation individuelle stable.

Dans ce cas, le référentiel doit déclarer que les axes sont latents et préciser le protocole ayant produit leur géométrie. L'ÉTR peut accueillir ces deux formes, mais elle ne doit pas les confondre.

15.17. Un référentiel comme espace de positionnement sémantique

Un référentiel sémantique peut être représenté par :

où :

- (\mathcal V_s) est le vocabulaire ou l'ensemble des objets

admissibles ;

- (\mathcal K_s) est l'architecture des coordonnées ;
- (\kappa_s) est la fonction de positionnement ;
- (\rho_s) est la relation utilisée ;
- (\mathcal F_s) contient les transformations autorisées ;
- (\mathcal B_s) définit les frontières ;
- (\Pi_s) conserve la provenance.

Le référentiel devient ainsi une structure complète de lecture et de navigation. Il contient à la fois :

- les éléments pouvant être positionnés ;
- les règles de positionnement ;
- les relations permettant de les comparer ;
- les transformations permettant de circuler entre eux.

15.18. Plusieurs architectures de coordonnées

Deux référentiels peuvent décrire les mêmes objets au moyen de coordonnées différentes. Pour un même objet (x) :

et :

avec :

Par exemple, une publication peut être positionnée dans :

é

selon son domaine et ses concepts ; dans :

selon l'objectif exprimé ; dans :

é

selon sa relation au passé, au présent ou au futur ; dans :

selon les comportements qu'elle provoque. Le contenu source reste identique. La structure de coordonnées active change.

15.19. Référentiel actif

Tous les référentiels disponibles ne doivent pas nécessairement être actifs simultanément. On définit un ensemble :

et un état d'activation :

où :

signifie que (R_i) est actif, et :

qu’il est inactif pour l’opération courante. L’ensemble actif est :

La désactivation d’un référentiel ne signifie pas sa suppression. Elle signifie que ses règles, ses coordonnées et ses relations ne participent pas à la lecture ou à la décision courante.

15.20. Sélection d’un référentiel

Un sélecteur peut choisir le référentiel pertinent :

où :

- (x) est l’entrée ;
- (C) est le contexte ;
- (\Omega) est la mémoire ;
- (G) est l’objectif.

Exemples :

é

é é

Le même objet est lu différemment selon la mission active.

15.21. Désactivation au profit d’un autre référentiel

Le système peut effectuer une transition :

Cette opération comporte deux décisions distinctes :

et :

Elle ne doit pas être confondue avec une transformation directe des coordonnées. Trois cas sont possibles. Relecture depuis l’objet source

Le système reprend l’objet initial et le positionne dans le nouveau référentiel. Transport inter-référentiel

Cette opération n’est valide que si une correspondance explicite existe. Absence de correspondance Si aucune transformation admissible n’existe :

é

le système doit relire l’objet source ou s’abstenir. Il ne doit pas inventer une équivalence.

15.22. Référentiel principal et référentiels secondaires

Une opération peut utiliser un référentiel principal :

et plusieurs référentiels secondaires :

Le référentiel principal détermine :

- les coordonnées principales ;
- les relations décisionnelles ;
- les frontières ;
- la politique de routage.

Les référentiels secondaires peuvent fournir :

- des qualifications ;
- des contrôles ;
- des contradictions ;
- des informations complémentaires.

Exemple :

é

avec :

Le référentiel linguistique aide à lire la demande, mais ne gouverne pas la décision médicale.

15.23. Conflit entre référentiels

Deux référentiels peuvent produire des orientations différentes.

Ce conflit ne constitue pas nécessairement une erreur. Il peut révéler que les référentiels poursuivent des objectifs différents. Par exemple :

peut favoriser la rapidité, tandis que :

favorise la conformité. Le système doit alors déclarer :

- quel référentiel est prioritaire ;
- pourquoi ;
- pour quelle durée ;
- avec quelle autorité ;
- quelles conclusions secondaires sont conservées.

15.24. Activation contextuelle et stabilité

Un changement de référentiel peut modifier fortement le résultat. Le système doit donc éviter une commutation arbitraire. La sélection peut être soumise à :

- un seuil de confiance ;
- une règle de priorité ;
- une validation humaine ;
- une durée minimale d'activation ;
- un coût de commutation ;
- une condition de sortie.

On peut écrire :

uniquement si :

Cette discipline empêche le système de choisir après coup le référentiel qui justifie le résultat désiré.

15.25. Registre d’activation

Chaque changement doit être conservé dans la provenance :

è

Le système doit pouvoir répondre :

- quel référentiel était actif ?
- à quel moment ?
- pourquoi a-t-il été remplacé ?
- quelles coordonnées ont été recalculées ?
- quelles informations ont été conservées ?
- quelles décisions ont changé ?

Le changement de référentiel devient ainsi auditable.

15.26. Exemple sémantique

Considérons la phrase : Cette stratégie est agressive. Dans un référentiel commercial :

« agressive » peut être qualifié comme :

- conquête ;
- croissance rapide ;
- intensification concurrentielle.

Dans un référentiel comportemental :

le même terme peut évoquer :

- hostilité ;
- conflit ;
- risque relationnel.

Dans un référentiel financier :

il peut signifier :

- forte exposition ;
- investissement élevé ;
- risque accru.

Le terme source ne change pas. L'architecture de coordonnées et les relations disponibles changent. Le système ne doit donc pas fusionner immédiatement ces trois lectures. Il doit identifier le référentiel actif.

15.27. Exemple pour Cercle

Une publication peut être positionnée dans plusieurs référentiels de pintones :

é

é

Le moteur peut activer :

é

pour rechercher les publications proches de la LOI de l'utilisateur. Il peut ensuite le désactiver et activer :

afin d'introduire une distance contrôlée. Enfin :

é

peut intervenir comme référentiel de contrôle sans déterminer la navigation principale. Ainsi, le choix d'une publication ne dépend pas d'un espace sémantique universel unique. Il dépend du référentiel et de l'objectif actifs.

15.28. Schéma général enrichi

Le fonctionnement devient :

Puis, si l'objectif change :

Le changement d'objectif entraîne donc un changement déclaré de référentiel, et potentiellement une nouvelle position de l'objet.

Conclusion révisée du chapitre 15

Un référentiel n'est pas seulement un contexte abstrait. Il peut contenir une architecture complète de coordonnées dans laquelle les objets sont positionnés et reliés. Dans le Référentiel Relationnel Unitaire :

et, en plusieurs dimensions :

L'unité :

est l'identité relationnelle. L'échelle décrit des facteurs de transformation et non des distances additives uniformes. Chaque référentiel peut définir :

- ses objets ;
- ses dimensions ;
- ses coordonnées ;
- ses relations sémantiques ;
- ses transformations ;
- ses frontières ;
- ses décisions admissibles.

Plusieurs référentiels peuvent coexister, mais ils ne doivent pas nécessairement être actifs simultanément. Le système peut :

é é é é

sans supprimer les autres référentiels ni fusionner leurs coordonnées. La formulation centrale du chapitre devient donc :

è à é é

Et le changement de référentiel constitue lui-même une transformation traçable, soumise à des conditions de validité.

Conclusion

Un référentiel n'est ni une base de données, ni un modèle d'intelligence artificielle, ni un ensemble de règles métier. Il constitue le langage commun dans lequel les objets, les transformations et les relations acquièrent une signification. Dans l'ÉTR, aucune relation ne devrait être interprétée sans que son référentiel soit explicitement identifié.

é é é é é é é

Chapitre 16 — Relations et mesures : comparer sans réduire selon le temps

Après la lecture, Compose, la qualification et séquence de sélection du référentiel, le système doit comparer des états. Cette comparaison ne doit pas être confondue avec la décision. Elle répond à une question précise : Quelle relation existe entre deux états comparables dans un référentiel donné ? La forme générale est :

où :

- (\mathcal Q_R) est l’espace des états du référentiel (R) ;
- (\mathcal L_R) est l’espace des relations admissibles ;
- (\rho_R(Q_i,Q_j)) est la relation de (Q_i) vers (Q_j).

16.1. La comparaison exige une comparabilité préalable

Avant tout calcul, le système vérifie :

Cette vérification porte au minimum sur :

- le type ;
- le référentiel ;
- la granularité ;
- la temporalité ;
- les unités ;
- la provenance ;
- les dimensions disponibles.

La comparaison n’est autorisée que si :

Si la comparabilité est partielle, les dimensions non comparables doivent être explicitement exclues ou marquées comme indéterminées.

16.2. Relation, distance et similarité

Ces trois notions doivent être distinguées. Relation Une relation décrit le rapport entre deux états.

Distance Une distance mesure une séparation.

Une distance peut satisfaire :

si elle est métrique. Similarité Une similarité mesure une proximité ou une compatibilité.

Elle peut être bornée, par exemple :

Une relation n’est pas nécessairement une distance. Une distance n’est pas nécessairement une similarité. Une similarité n’est pas nécessairement symétrique.

16.3. Relation orientée

L'ÉTR distingue :

de :

Dans un régime multiplicatif :

—

et :

—

donc :

La relation possède une orientation. Elle décrit un passage de source vers cible. Cette orientation disparaît si l'on remplace trop tôt la relation par une distance symétrique.

16.4. Relation élémentaire multiplicative

Pour deux valeurs strictement positives :

la relation est :

—

Interprétation :

indique une expansion ;

indique l'identité ;

indique une contraction relative. Exemples :

—

16.5. Relation multidimensionnelle

Pour :

et :

avec toutes les composantes strictement positives :

— —

Le résultat est un profil relationnel. Chaque composante conserve son sens propre. Pour :

on obtient :

Cette forme conserve :

- la transformation forte sur (x) ;
- l'identité sur (y) ;
- la transformation modérée sur (z).

16.6. Le profil relationnel

Dans un référentiel sémantique, la relation peut contenir plusieurs dimensions hétérogènes :

é é é é

Le profil relationnel précède l'agrégation. Il permet de conserver des combinaisons telles que :

é

et simultanément :

Une valeur globale unique masquerait cette structure.

16.7. Mesure par dimension

Chaque dimension peut avoir sa propre fonction :

Le profil complet devient :

Les dimensions peuvent être :

- numériques ;
- ordinales ;
- symboliques ;
- booléennes ;
- probabilistes ;
- relationnelles.

Le système ne doit pas les forcer dans une même échelle sans transformation explicite.

16.8. Agrégation

Une agrégation transforme un profil en résultat compact :

Exemple pondéré :

avec :

Cette opération doit déclarer :

- les poids ;
- la normalisation ;
- les dimensions exclues ;
- les valeurs manquantes ;
- le domaine d'usage ;
- les pertes produites.

L'agrégation n'est pas la relation. Elle est une transformation appliquée à la relation.

16.9. Risque de compensation

Une agrégation additive peut produire une compensation. Supposons :

et :

Le même résultat peut être produit par :

Les deux profils sont pourtant différents.

Une agrégation peut donc rendre plusieurs structures relationnelles indiscernables.

16.10. Veto et contrainte dure

Certaines dimensions ne doivent pas être compensées. Exemple :

é é é

Même avec une forte proximité sémantique, la décision peut être interdite. On distingue alors : Signal pondérable

peut participer à une agrégation. Contrainte dure

doit être satisfaite. La fonction de validité devient :

puis seulement, si elle est vraie :

16.11. Relation et contexte

Une relation peut dépendre du contexte :

Le même couple peut être lu différemment selon :

- la tâche ;
- l'utilisateur ;
- le temps ;
- l'environnement ;
- le risque ;
- l'objectif.

Mais le contexte doit être déclaré. Sinon, deux calculs différents paraissent contradictoires alors qu'ils répondent à des conditions différentes.

16.12. Relation et provenance

La relation doit conserver la provenance de ses entrées et de sa méthode.

é

Deux relations numériquement identiques peuvent ne pas avoir la même qualité si elles proviennent :

- de sources différentes ;
- de modèles différents ;
- de versions différentes ;
- de référentiels différents.

16.13. Relation et incertitude

Une relation peut être accompagnée d’une incertitude :

Exemple :

é

Mais cette écriture n’est valide que si le sens de (U_\rho) est défini.

Il peut représenter :

- une variance ;
- un intervalle ;
- une confiance ;
- une erreur estimée ;
- une qualité de source.

16.14. Relation inverse

Dans un régime inversible :

Pour le rapport multiplicatif :

— —

En plusieurs dimensions :

avec inversion composante par composante. Cette propriété doit être testée lorsque le système la revendique.

16.15. Composition des relations

Pour trois états :

un régime multiplicatif cohérent satisfait :

où (\odot) est le produit composante par composante.

En dimension unique :

— —

Cette propriété permet de composer des transports. Elle ne doit pas être étendue automatiquement à toutes les relations sémantiques.

16.16. Relation directe et relation composée

Deux calculs peuvent être comparés :

et :

é

L'écart :

é

mesure une cohérence de composition. Si :

la composition est cohérente dans le cadre déclaré. Sinon, le système doit rechercher :

- une perte ;
- une dépendance au chemin ;
- une qualification différente ;
- une erreur ;
- un changement de référentiel.

16.17. Relations non composables

Deux relations ne sont composables que si leurs signatures sont compatibles.

Alors :

Mais si :

et :

avec :

la composition n’est pas définie sans médiation. Le typage précède donc la composition.

16.18. Mesures relationnelles possibles

L’ÉTR n’impose pas une métrique universelle. Selon le référentiel, on peut utiliser :

- rapport multiplicatif ;
- distance euclidienne ;
- similarité cosinus ;
- distance de graphe ;
- relation ordinale ;
- compatibilité logique ;
- divergence probabiliste ;
- écart temporel ;
- transformation symbolique.

La méthode doit appartenir au référentiel :

16.19. Relation sémantique explicite

Une relation sémantique peut être typée :

é

Exemple :

Le type « cause probable » ne doit pas être confondu avec une simple proximité. Deux objets proches ne sont pas nécessairement causalement liés.

16.20. Relation sémantique issue d’un modèle

Un LLM ou un embedding peut proposer une relation :

è

Cette relation reste une estimation. Elle doit contenir :

- le modèle ;
- le prompt ou protocole ;
- la version ;
- la sortie brute ;
- la qualification retenue ;
- la validation éventuelle.

L’ÉTR n’élève pas automatiquement une estimation de modèle au statut de relation validée.

16.21. Comparaison entre référentiels

Deux référentiels peuvent produire :

et :

Ces relations ne doivent pas être directement comparées si leurs codomaines diffèrent. Il faut une transformation :

ou un espace commun :

Sans cela, la comparaison inter-référentielle est indéfinie.

16.22. Référentiel désactivé

Lorsqu’un référentiel est désactivé :

ses relations ne doivent plus contribuer au calcul courant. Les résultats antérieurs restent toutefois en mémoire :

Le système doit distinguer :

- relation active ;
- relation historique ;
- relation invalidée ;
- relation suspendue.

16.23. Exemple documentaire

Requête : responsabilité d'un dirigeant en cas de faute de gestion Document (D_1) :

- très proche lexicalement ;
- ancien ;
- juridiction différente.

Document (D_2) :

- moins proche lexicalement ;
- récent ;
- juridiction correcte.

Profils :

avec les dimensions :

é é é

Le document (D_1) n'est pas automatiquement préférable. Le routeur décidera selon le référentiel juridique.

16.24. Exemple Cercle

Une publication peut être comparée à une LOI selon :

é é é

Dans le référentiel de proximité, le système privilégie :

é

Dans le référentiel d’exploration, il peut privilégier :

é

sous contrainte de :

é

Le profil ne change pas. La politique d’exploitation change.

16.25. Sortie de la comparaison

La sortie complète doit être :

où :

- (\rho) est le profil relationnel ;
- (R) est le référentiel ;
- (\Pi) est la provenance ;
- (U) est l’incertitude ;
- (C) est le rapport de comparabilité ;
- (V) est le rapport de validation.

Cette sortie est transmise au routeur. Elle ne constitue pas encore une décision.

Conclusion du chapitre

Comparer dans l'ÉTR consiste à produire une relation typée entre des états comparables dans un référentiel déclaré. La relation peut être :

- orientée ;
- multidimensionnelle ;
- multiplicative ;
- sémantique ;
- logique ;
- temporelle ;
- probabiliste.

Le principe central est :

é é

La comparaison décrit. L'agrégation compresse. Le routeur décidera. Le chapitre suivant peut donc traiter précisément le routeur, c'est-à-dire la transformation qui convertit un profil relationnel en orientation, sélection, abstention ou action proposée.

Chapitre 17 — La mémoire : conserver les états, les relations et leurs transformations

Dans l'ÉTR, la mémoire ne se réduit pas à un espace de stockage. Elle conserve :

- les objets ;
- les états ;
- les relations ;
- les transformations ;
- les chemins ;
- les référentiels ;
- les provenances ;
- les contradictions ;
- les décisions.

Sa fonction centrale est la suivante :

é é é é

17.1. Définition générale

La mémoire relationnelle est notée :

Une forme minimale est :

où :

- (V) contient les objets et les états ;
- (E) contient les relations et les transformations ;
- (\Gamma) contient les chemins ;
- (\mathfrak R) contient les référentiels ;
- (\Pi) contient les provenances.

Cette mémoire est donc à la fois :

- un registre d'états ;
- un graphe relationnel ;
- un historique de transformations ;
- un système de versionnement.

17.2. Mémoire de données et mémoire relationnelle

Une mémoire classique peut conserver :

ou :

Une mémoire relationnelle conserve plutôt :

où :

- (x) est l’état initial ;
- (T) est la transformation ;
- (y) est l’état produit ;
- (R) est le référentiel ;
- (\Pi) est la provenance ;
- (U) est l’incertitude.

Deux résultats identiques peuvent ainsi rester distincts s’ils ont été produits par des chemins différents.

17.3. Mémoire paramétrique, contextuelle et relationnelle

Ces trois formes doivent être distinguées. Mémoire paramétrique Elle est contenue dans les paramètres d’un modèle. Exemple :

Elle est :

- distribuée ;
- compressée ;
- difficile à modifier ponctuellement ;
- difficile à attribuer à une source précise.

Mémoire contextuelle Elle contient les informations disponibles durant l’inférence. Exemple :

Elle est :

- temporaire ;
- limitée ;
- dépendante de la session ;
- sensible à l’ordre.

Mémoire relationnelle Elle conserve explicitement :

Elle est :

- persistante ;
- versionnée ;
- inspectable ;
- modifiable par transformations déclarées.

Ces mémoires peuvent coopérer sans être confondues.

17.4. Unité mémorielle minimale

Une unité mémorielle peut être définie par :

où :

- (id) est l'identifiant ;
- (x) est l'objet source ;
- (Q) est son état relationnel ;
- (R) est le référentiel actif ;
- (\Pi) est la provenance ;
- (U) est l'incertitude ;
- (t) est la temporalité.

Si l'état provient d'une transformation :

La mémoire conserve alors le lien entre les versions.

17.5. La mémoire comme graphe orienté

La mémoire peut être représentée par un graphe :

Les nœuds représentent :

- des objets ;
- des états ;
- des décisions ;
- des référentiels.

Les arêtes représentent :

- des relations ;
- des transformations ;
- des mises à jour ;
- des dépendances ;
- des contradictions.

Exemple :

La relation entre les deux états reste distincte du transport :

sauf si le régime considéré les identifie explicitement.

17.6. Update

La mémoire évolue par une transformation déclarée :

où (\Delta_t) est la modification proposée.

Une forme plus développée est :

La mise à jour peut être déclenchée par :

- une nouvelle observation ;
- une qualification ;
- une décision ;
- une correction ;
- un changement de référentiel ;
- une validation humaine.

17.7. Update ne signifie pas seulement ajouter

Une mise à jour peut produire plusieurs effets :

où :

- (\Delta^+) : éléments ajoutés ;
- (\Delta^\sim) : éléments modifiés ;
- (\Delta^-) : éléments retirés de l'état actif ;
- (\Delta^?) : éléments devenus indéterminés ;
- (\Delta^!) : conflits ou alertes créés.

L'ancienne information ne doit pas être effacée sans déclaration.

17.8. Mémoire croissante

Une mémoire croissante conserve l'histoire des transformations. Si :

la mémoire garde :

Elle ne remplace pas simplement :

par :

La croissance concerne donc la traçabilité, non l'accumulation aveugle. Une information peut devenir inactive tout en restant historiquement accessible.

17.9. État actif et état historique

La mémoire distingue :

de :

Un état historique peut être :

- remplacé ;
- obsolète ;
- invalidé ;
- suspendu ;
- contradictoire.

Mais il reste conservé avec son statut. Exemple :

é

La mémoire doit permettre de comprendre pourquoi (Q_1) n'est plus actif.

17.10. Versionnement

Chaque objet mémorisé peut recevoir une suite de versions :

avec :

Le registre de version conserve :

- l'état précédent ;
- la transformation ;
- l'auteur ;
- la date ;
- le référentiel ;
- la justification ;
- les effets informationnels.

Le dernier état n'annule pas l'histoire.

17.11. Provenance

La provenance est obligatoire.

é

Une donnée sans provenance peut être stockée, mais elle ne doit pas être traitée comme équivalente à une donnée vérifiée. La mémoire doit distinguer :

de :

é

17.12. Provenance transitive

Lorsqu'une information traverse plusieurs agents ou modèles, la provenance ne doit pas être remplacée par celle du dernier intermédiaire. Si :

la provenance finale doit conserver la chaîne :

Trois agents répétant la même source ne constituent pas trois sources indépendantes.

17.13. Contradictions

La mémoire peut contenir plusieurs assertions incompatibles.

et :

Le système ne doit pas nécessairement supprimer immédiatement l'une d'elles. Il peut créer un objet conflit :

Le conflit peut recevoir un statut :

- ouvert ;
- contextualisé ;
- résolu ;
- non résoluble ;
- soumis à validation.

17.14. Ne pas confondre contradiction et changement de référentiel

Deux relations peuvent diverger parce qu'elles appartiennent à deux référentiels différents :

Ce n'est pas nécessairement une contradiction. La contradiction n'existe que si les assertions sont incompatibles dans un même cadre de validité. Il faut donc comparer :

et :

avant de conclure à un conflit.

17.15. Mémoire et référentiels actifs

Chaque référentiel peut posséder une mémoire locale :

La mémoire globale devient :

avec conservation des frontières entre référentiels. Lorsqu'un référentiel est désactivé :

sa mémoire n’est pas supprimée. Elle cesse seulement d’alimenter le calcul courant.

17.16. Activation mémorielle

Une requête ne doit pas charger toute la mémoire. Une fonction d’activation sélectionne un sous-ensemble :

où :

- (Q) est l’état courant ;
- (R) est le référentiel actif ;
- (G) est l’objectif ;
- (C) est le contexte.

La mémoire totale reste :

mais seule une partie devient active :

17.17. Rappel et recherche

La mémoire peut être interrogée par :

- identifiant ;
- relation ;
- chemin ;
- référentiel ;
- temporalité ;
- provenance ;
- proximité ;
- statut.

Exemples :

La recherche mémorielle n'est pas limitée à une similarité vectorielle.

17.18. Mémoire et Compose

Compose peut utiliser un état mémoriel :

puis construire :

La mémoire influence donc l'état initial.

Mais Compose ne doit pas modifier automatiquement (\Omega).

La mise à jour reste une transformation distincte :

17.19. Mémoire et qualification

Une qualification peut dépendre de la mémoire :

Exemple : le terme « Java » peut être qualifié différemment selon que la mémoire active concerne :

- la programmation ;
- l'Indonésie ;
- le café.

La mémoire aide à sélectionner le référentiel et la qualification. Elle ne constitue pas une preuve de leur validité.

17.20. Mémoire et décision

Le routeur peut consulter :

pour déterminer :

- des précédents ;
- des interdictions ;
- des préférences ;
- des seuils ;
- des conséquences observées.

La décision devient :

Une mémoire inadéquate peut donc orienter une décision incorrecte. La sélection mémorielle doit elle aussi être traçable.

17.21. Écriture conditionnelle

Toute sortie ne doit pas nécessairement être inscrite en mémoire durable. Une condition d'écriture peut être définie :

La mémoire peut refuser l'écriture lorsque :

- la provenance est insuffisante ;
- l'incertitude est trop élevée ;
- le résultat est temporaire ;
- le référentiel interdit la persistance ;
- une validation est requise ;
- l'information est sensible.

17.22. Mémoire temporaire et durable

On distingue : Mémoire de travail

Elle conserve les états nécessaires au calcul courant. Mémoire de session

Elle persiste pendant une interaction ou une mission. Mémoire durable

Elle conserve les objets validés au-delà de la session. Le passage de l'une à l'autre constitue une transformation :

Cette consolidation doit être contrôlée.

17.23. Oubli et suppression

Une mémoire rigoureuse doit permettre l'oubli. Mais plusieurs opérations doivent être distinguées. Désactivation

L'élément ne participe plus au calcul. Archivage

L'élément reste consultable historiquement. Suppression logique

L'élément devient inaccessible aux usages ordinaires. Suppression physique

Les données sont effectivement retirées. Ces opérations ne sont pas équivalentes.

17.24. Compression de mémoire

Une mémoire exhaustive devient coûteuse. Une opération de compression peut produire :

Elle peut :

- agréger des chemins similaires ;
- résumer des états ;
- supprimer des redondances ;
- créer des prototypes ;
- conserver des points de contrôle.

Mais elle doit déclarer les informations devenues indéterminables. Une mémoire compressée ne doit pas être présentée comme identique à l'historique complet.

17.25. Prototype et membre réel

Un référentiel peut être représenté provisoirement par un prototype :

Ce prototype peut être :

- un état représentatif ;
- un membre réel ;
- un médoïde ;
- une synthèse validée.

Il sert à accélérer la comparaison. Mais il ne remplace pas l'ensemble des membres du référentiel.

Il constitue une lecture compacte du référentiel.

17.26. Mémoire non destructive

Une mise à jour non destructive conserve les états antérieurs.

Cette formule doit être nuancée : l'union ne suffit pas toujours, car certaines relations changent de statut. Une forme plus précise est :

Le système ajoute les nouveaux objets et met à jour les statuts sans effacer l'histoire.

17.27. Validation d'Update

Une mise à jour est valide si :

Les contrôles peuvent porter sur :

- les types ;
- les permissions ;
- la provenance ;
- les contradictions ;
- les invariants ;
- les frontières ;
- la cohérence temporelle.

Si la mise à jour échoue :

et le système conserve un rapport d'échec.

17.28. Transactions

Une mise à jour comportant plusieurs opérations doit être atomique lorsque cela est nécessaire.

Deux résultats sont alors admissibles :

é é

ou :

é

Cette propriété évite les mémoires partiellement mises à jour.

17.29. Permissions

L'écriture peut dépendre de l'acteur :

Les droits peuvent inclure :

- lire ;
- proposer ;
- qualifier ;
- écrire ;
- corriger ;
- invalider ;
- supprimer ;
- valider.

Un LLM peut proposer une mise à jour sans avoir le droit de la consolider.

17.30. Exemple documentaire

Un article affirme : Le procédé réduit la consommation de 30 %. La mémoire conserve :

é é

avec :

Une étude ultérieure mesure :

é é

La mémoire ne remplace pas simplement (30%) par (12%). Elle conserve :

é

avec les protocoles, dates et incertitudes correspondants.

17.31. Exemple pour Cercle

Pour un utilisateur, la mémoire peut conserver :

- les pintones consultés ;
- les LOI successives ;
- les transitions ;
- les référentiels activés ;
- les publications sélectionnées ;
- les exclusions ;
- les changements de direction.

On peut écrire :

Le système ne conserve pas seulement les contenus aimés. Il conserve les chemins relationnels ayant conduit à leur activation.

17.32. Changement de référentiel et mémoire

Lorsqu'un référentiel change :

la mémoire doit conserver :

Les états du référentiel précédent restent disponibles :

mais cessent d'être actifs si :

Les nouveaux états sont recalculés ou transportés dans :

17.33. Sortie d'Update

La sortie complète d'une mise à jour est :

é é

où :

- (\Omega_{t+1}) est la mémoire obtenue ;
- (\Delta_{\text{appliqué}}) contient les modifications validées ;
- (\Delta_{\text{refusé}}) contient les modifications rejetées ;
- (C) contient les conflits ;
- (\Pi) contient la provenance ;
- (V) est le rapport de validation.

Conclusion du chapitre

La mémoire de l'ÉTR n'est pas un simple réservoir de données. Elle conserve :

é

Sa transformation centrale est :

Toute mise à jour doit préciser :

- ce qui est ajouté ;
- ce qui est modifié ;
- ce qui devient inactif ;
- ce qui est contesté ;
- ce qui reste indéterminé ;
- ce qui est conservé historiquement.

La mémoire permet ainsi au système de ne pas seulement produire des décisions, mais d'apprendre sans effacer les conditions de cet apprentissage.

Chapitre 18 — Le routeur : transformer un profil relationnel en décision

La comparaison produit une relation. Le routeur détermine ce qu'il faut en faire. Cette distinction est fondamentale :

é é

Une forte proximité, une forte cohérence ou une forte distance ne constituent pas encore une décision. Le routeur applique une politique déclarée à un profil relationnel, dans un référentiel donné, avec des contraintes, une mémoire et un niveau de risque.

18.1. Définition générale

Le routeur est une transformation :

où :

- (\mathcal L_R) est l'espace des relations ;
- (R) est le référentiel actif ;
- (\Omega) est la mémoire ;
- (C) contient les contraintes ;
- (U) contient les incertitudes ;
- (\mathcal D_R) est l'espace des décisions admissibles.

La forme opérationnelle est :

18.2. Entrée du routeur

Le routeur ne doit pas recevoir uniquement un score.

Il reçoit idéalement un objet complet :

avec :

- (\rho) : profil relationnel ;
- (R) : référentiel ;
- (\Pi) : provenance ;
- (U) : incertitude ;
- (C_{\text{comp}}) : comparabilité ;
- (V) : validation.

Le routeur doit donc pouvoir refuser de décider si l'entrée est insuffisante.

18.3. La décision n'est pas contenue dans la relation

Supposons :

é

Cette valeur peut conduire à plusieurs décisions. Dans un moteur de recherche :

Dans un moteur d'exploration :

é é

Dans un système de détection de fraude :

La relation reste identique. Le référentiel et l'objectif changent.

18.4. Décisions admissibles

Le routeur peut produire :

é é é

Selon l'application, il peut aussi produire :

- un agent à solliciter ;
- un référentiel à activer ;
- une source à consulter ;
- une action à proposer ;
- un niveau de priorité.

18.5. Politique de routage

Le routeur applique une politique :

On écrit :

La politique doit être distincte du référentiel, même si elle lui est rattachée. Le référentiel définit :

- les objets ;
- les relations ;
- les transformations admissibles.

La politique définit :

- les priorités ;
- les seuils ;
- les arbitrages ;
- les sorties.

18.6. Routeur déterministe

Un routeur déterministe produit toujours la même décision pour les mêmes entrées.

Exemple :

é é

Avantages :

- simplicité ;
- reproductibilité ;
- audit facile.

Limite :

- faible souplesse.

18.7. Routeur probabiliste

Un routeur probabiliste produit une distribution :

puis sélectionne :

Ce régime peut être utile pour :

- l'exploration ;
- la diversification ;
- l'apprentissage ;
- la recommandation.

Mais la probabilité de décision ne doit pas être confondue avec la confiance dans la relation.

18.8. Routeur par règles

Une politique peut être décrite par des règles. Exemple :

é é é é é

Les règles doivent être :

- ordonnées ;
- typées ;
- non contradictoires ;
- versionnées.

18.9. Routeur multi-signal

Le routeur peut utiliser plusieurs dimensions :

Il peut appliquer :

é é é é

Le routeur ne doit pas obligatoirement réduire ces signaux en une moyenne unique. Il peut conserver des règles conditionnelles.

18.10. Contraintes dures

Certaines conditions interdisent une décision.

La validité est :

Si :

le routeur ne doit pas compenser ce résultat par d'autres signaux favorables. Exemple :

é é é

implique :

même si :

é

est élevé.

18.11. Seuils

Un seuil transforme une mesure continue en décision discrète.

Le seuil doit être déclaré avec :

- son origine ;
- son domaine ;
- sa version ;
- sa méthode de calibration ;
- ses conséquences.

Un seuil arbitraire ne constitue pas une politique scientifique.

18.12. Zone d'incertitude

Un système rigoureux peut définir une zone intermédiaire :

Dans cette zone :

é

Ainsi, le routeur ne force pas une décision binaire lorsque la relation est ambiguë.

18.13. Abstention

L'abstention est une décision valide.

Elle peut être produite lorsque :

- la comparabilité est insuffisante ;
- l'incertitude est trop élevée ;
- la provenance est faible ;
- les référentiels sont en conflit ;
- aucune politique ne s'applique ;
- le risque est excessif.

L'abstention protège le système contre la décision forcée.

18.14. Vérification

Le routeur peut produire :

Cette décision peut déclencher :

- une recherche documentaire ;
- un second modèle ;
- un expert humain ;
- un calcul supplémentaire ;
- un changement de référentiel.

Le routeur devient alors un orchestrateur.

18.15. Sélection d'un référentiel

Le routeur peut décider que le référentiel actuel n'est pas adapté.

Cette décision doit déclarer :

- la cause ;
- le référentiel désactivé ;
- le référentiel activé ;
- les données à relire ;
- les relations à recalculer.

Le changement de référentiel ne doit pas être implicite.

18.16. Sélection d'un agent

Dans un système multi-agent :

Le choix peut dépendre de :

é û é

Le routeur ne choisit donc pas seulement une réponse. Il peut choisir le producteur de la prochaine transformation.

18.17. Décision et action

Une décision n'est pas nécessairement une action.

La décision doit parfois être compilée :

Exemple :

doit devenir :

é

La compilation possède son propre contrat.

18.18. Validation humaine

Dans un système sensible :

Le routeur peut produire :

à

L'humain reste l'autorité finale. Cette séparation est importante en :

- médecine ;
- droit ;
- finance ;
- robotique ;
- administration.

18.19. Routeur et mémoire

Le routeur peut consulter :

afin d'utiliser :

- des précédents ;
- des interdictions ;
- des préférences ;
- des décisions antérieures ;
- des conséquences observées.

La décision devient :

La mémoire activée doit être enregistrée dans la provenance.

18.20. Routeur sans mise à jour implicite

Le routeur produit une décision. Il ne doit pas modifier directement la mémoire sauf si cette fonction est explicitement déclarée. Régime séparé :

puis :

Cette séparation permet de contrôler l'écriture.

18.21. Routeur hiérarchique

Plusieurs routeurs peuvent être organisés en niveaux.

Exemple : . routeur de sécurité ;

. routeur de domaine ;

. routeur de pertinence ;

. routeur d'action.

Le premier peut bloquer les suivants.

é é ê

Cette architecture évite qu'une décision commerciale compense une contrainte de sécurité.

18.22. Routeur parallèle

Plusieurs politiques peuvent produire des décisions :

Un arbitre produit :

Cette architecture permet de comparer :

- politique juridique ;
- politique financière ;
- politique éthique ;
- politique opérationnelle.

Les divergences doivent être conservées.

18.23. Conflit de décisions

Si :

le système crée un conflit :

Le conflit peut être :

- arbitré ;
- différé ;
- soumis à validation ;
- conservé.

Le routeur ne doit pas masquer cette divergence par une moyenne.

18.24. Priorité

Une politique peut définir une relation de priorité :

é é

Cela signifie que la sécurité domine la performance en cas de conflit. La priorité doit être :

- explicite ;
- versionnée ;
- liée à un domaine ;
- justifiée.

18.25. Coût

Une décision peut être évaluée par son coût :

Le routeur peut chercher :

sous contraintes :

Le coût peut inclure :

- temps ;
- énergie ;
- argent ;
- risque ;
- ressources ;
- impact social.

18.26. Utilité

Une politique peut utiliser une fonction d'utilité :

et choisir :

Mais la fonction d'utilité doit être distincte de l'incertitude (U). La notation doit éviter toute ambiguïté.

18.27. Exploration et exploitation

Le routeur peut arbitrer entre : Exploitation Choisir l'option la plus proche ou la plus performante connue. Exploration Choisir une option plus distante pour acquérir de l'information ou diversifier. Une politique simple peut être :

é é

Dans l'ÉTR, la distance exploratoire peut être contrainte par un référentiel actif.

18.28. Exemple documentaire

Profil :

avec :

é é é

Politique :

é

mais :

é

Décision :

é

Le document n’est ni accepté ni rejeté immédiatement.

18.29. Exemple pour Cercle

Profil d’une publication :

é é é

Dans le référentiel de proximité :

si :

é

Dans le référentiel d’exploration :

si :

é

et :

é

Le même profil peut donc être exploité différemment.

18.30. Exemple robotique

Un robot détecte un obstacle. Profil :

Le routeur peut produire :

ê

La décision est ensuite compilée en commande.

18.31. Provenance de la décision

Toute décision doit conserver :

Le système doit pouvoir répondre :

- quelle relation a été utilisée ?
- quel référentiel était actif ?
- quelle politique a été appliquée ?
- quelle mémoire a été consultée ?
- quelles contraintes ont été déclenchées ?
- quelle incertitude subsistait ?

18.32. Explication de décision

Une explication peut être produite à partir du registre :

Cette explication repose sur les transformations enregistrées. Elle ne doit pas être remplacée par une justification générée sans accès aux causes effectives. Un LLM peut formuler l'explication, mais le contenu doit venir du registre décisionnel.

18.33. Test du routeur

Le routeur doit être testé sur :

- les seuils ;
- les cas limites ;
- les conflits ;
- les entrées incomplètes ;
- les référentiels désactivés ;
- les contraintes dures ;
- l'abstention ;
- la reproductibilité.

On vérifie notamment :

et :

pour de faibles variations de (x). Une forte instabilité doit être mesurée.

18.34. Sortie du routeur

La sortie complète est :

où :

- (D) est la décision ;
- (R) est le référentiel ;
- (\pi_R) est la politique ;
- (C) contient les contraintes activées ;
- (U) contient l'incertitude ;
- (\Pi) contient la provenance ;
- (V) est le rapport de validation.

Conclusion du chapitre

Le routeur transforme un profil relationnel en décision. Sa forme générale est :

Il ne crée pas la relation. Il ne doit pas modifier implicitement la mémoire. Il applique une politique déclarée à des relations déjà calculées. Le routeur permet ainsi de séparer :

è é é

Cette séparation rend possible :

- plusieurs politiques sur un même profil ;
- l’abstention ;
- la vérification ;
- le changement de référentiel ;
- l’arbitrage entre agents ;
- la validation humaine ;
- l’audit complet de la décision.

Je pense que c'est effectivement le chapitre central du livre. En prenant du recul sur tout ce que nous avons écrit, je dirais même ceci : Compose est le cœur mécanique de l'ÉTR. La Qualification est son cœur cognitif. C'est probablement ici que l'ÉTR se différencie le plus des architectures actuelles. Les LLM produisent directement une interprétation. Dans l'ÉTR, l'interprétation devient une transformation explicite, avec un contrat, une provenance et une possibilité d'audit. Je proposerais donc de traiter ce chapitre comme un véritable pilier du livre.

Chapitre XIX — La Qualification De la structure à la connaissance

19.1. Pourquoi Compose ne suffit pas

Compose construit un état. Il ne lui attribue pas encore de signification. Considérons la phrase : Le Jaguar accélère rapidement. Après lecture et Compose, le système connaît :

- l'ordre ;
- les dépendances ;
- les transformations successives ;
- le chemin de construction.

Mais une question demeure ouverte : De quel « Jaguar » s'agit-il ?

- d'un animal ;
- d'une automobile ;
- d'un logiciel ;
- d'une organisation ;
- d'un nom propre.

La structure est correcte. L'interprétation reste ouverte. C'est précisément le rôle de la qualification.

19.2. Définition

La qualification transforme un état structurel en un état interprété relativement à un référentiel.

où :

- (Q_{\mathrm{struct}}) est l'état construit par Compose ;
- (R) est le référentiel actif ;
- (\Omega) est la mémoire ;
- (C) représente le contexte courant.

Contrairement à Compose, la qualification n'est pas universelle. Elle dépend du cadre choisi.

19.3. La qualification n'est pas une vérité

Une qualification ne transforme jamais une hypothèse en vérité. Elle exprime : l'interprétation actuellement la plus cohérente avec le référentiel actif. Autrement dit,

é é

mais :

é é é

Cette distinction protège le système contre les affirmations implicites.

19.4. Les quatre piliers de la qualification

Toute qualification repose simultanément sur quatre éléments. La structure Ce que Compose a effectivement construit.

Le référentiel Le domaine dans lequel cette structure est lue.

La mémoire Les connaissances déjà disponibles.

Le contexte L'objectif poursuivi au moment de l'interprétation. Une modification de l'un de ces quatre éléments peut conduire à une qualification différente.

19.5. Une même structure, plusieurs qualifications

Considérons : Le Jaguar accélère rapidement. Structure :

Référentiel automobile :

é

Référentiel zoologique :

é

La structure est identique. La qualification change.

19.6. Les familles de qualification

L'ÉTR ne limite pas la qualification au langage. Elle peut être :

- sémantique ;
- logique ;
- scientifique ;
- juridique ;
- médicale ;
- stratégique ;
- comportementale ;
- économique ;
- pédagogique ;
- sociale.

Chaque famille possède son propre référentiel.

19.7. Qualification hiérarchique

Une qualification peut être construite progressivement. Mot ↓ expression ↓ phrase ↓ document ↓ raisonnement ↓ décision Chaque niveau enrichit le précédent sans le remplacer.

19.8. Qualification locale et globale

Une structure peut recevoir : une qualification locale :

puis une qualification globale :

Par exemple : chaque phrase d'un document est qualifiée individuellement, puis l'ensemble du document reçoit une qualification globale. Les deux niveaux ne doivent pas être confondus.

19.9. Qualification et mémoire

La mémoire influence la qualification. Mais elle ne doit jamais la déterminer automatiquement. Deux mémoires différentes peuvent produire deux qualifications différentes. La mémoire éclaire. Elle ne démontre pas.

19.10. Qualification et intelligence artificielle

Un modèle d'IA peut produire une qualification. Par exemple :

Dans l'ÉTR, cette sortie possède un statut précis : proposition de qualification. Elle doit pouvoir être :

- vérifiée ;
- remplacée ;
- corrigée ;
- contextualisée.

Le modèle n'est donc plus le détenteur de la connaissance. Il devient un producteur d'hypothèses.

19.11. Qualification humaine

Le même mécanisme vaut pour un expert.

Une qualification humaine possède également :

- une provenance ;
- un contexte ;
- un référentiel.

L'ÉTR place donc humain et IA sur un même plan méthodologique. La différence réside dans leur provenance, non dans leur statut logique.

19.12. Qualification collective

Plusieurs qualifications peuvent coexister. Le système peut conserver :

sans les fusionner immédiatement. Il devient possible :

- de rechercher un consensus ;
- d'identifier un conflit ;
- de maintenir plusieurs interprétations compatibles.

19.13. Qualification et incertitude

Toute qualification devrait être accompagnée de son niveau d'incertitude. Cette incertitude peut provenir :

- de la structure ;
- de la mémoire ;
- du référentiel ;
- du contexte ;
- du modèle ayant produit l'interprétation.

Ainsi :

La qualification devient un objet pleinement traçable.

19.14. Qualification et évolution

Une qualification peut évoluer. La structure initiale peut rester inchangée. La mémoire peut évoluer. Le référentiel peut évoluer. Le contexte peut évoluer. L'ÉTR autorise donc :

sans considérer cette évolution comme une erreur.

19.15. Contrat de qualification

Toute qualification devrait déclarer :

- les entrées utilisées ;
- le référentiel actif ;
- la mémoire consultée ;
- les hypothèses retenues ;
- les hypothèses rejetées ;
- les pertes éventuelles ;
- les invariants conservés ;
- la provenance ;
- l'incertitude.

Ainsi, une qualification devient une transformation scientifique à part entière.

19.16. Une architecture ouverte

L'un des apports majeurs de l'ÉTR est de considérer que la qualification n'appartient à aucune technologie particulière. Elle peut être produite par :

- un LLM ;
- un moteur symbolique ;
- un système expert ;
- un algorithme probabiliste ;
- un humain ;
- une combinaison de plusieurs agents.

Tous deviennent des moteurs de qualification, insérés dans une architecture commune. Le noyau de l'ÉTR reste inchangé.

Conclusion

Compose construit une structure. La qualification lui attribue une interprétation. La relation compare ces interprétations. Le routeur choisit une orientation. La mémoire conserve leur histoire. Cette séparation constitue l'un des principes fondateurs de l'ÉTR :

Chapitre XX — Validation scientifique, technique et protocoles d’évaluation De la proposition à l’épreuve Depuis le début de cet ouvrage, l’ÉTR a été présentée à travers ses objets fondamentaux : la lecture, Compose, les référentiels, la qualification, les

relations, la mémoire et le routeur. Ces objets ont été distingués, définis, articulés et rapportés aux fonctions qu’ils rendent possibles. Cette construction théorique ne suffit cependant pas à établir leur validité. Une architecture ne devient pas scientifique parce qu’elle possède un vocabulaire cohérent, parce qu’elle peut être implémentée ou parce qu’elle produit quelques résultats favorables. Elle devient scientifiquement examinable lorsque les affirmations qui la composent sont suffisamment précises pour être confrontées à des observations, à des démonstrations ou à des contre-exemples susceptibles de les soutenir, de les limiter, de les réviser ou de les contredire. La question :

Ė

est donc trop générale pour recevoir une réponse scientifique. Elle réunit sous une même formulation plusieurs problèmes distincts. Les objets de l’ÉTR sont-ils définis sans ambiguïté ? Les transformations qui leur sont attribuées respectent-elles les propriétés annoncées ? Les algorithmes proposés réalisent-ils effectivement ces transformations ? Le logiciel implémente-t-il fidèlement les algorithmes décrits ? Les performances observées proviennent-elles des mécanismes revendiqués ? Ces performances demeurent-elles lorsque les données, les référentiels, les paramètres ou les contraintes changent ? L’architecture répond-elle à un besoin réel ? Peut-elle être qualifiée pour un domaine d’usage déterminé ? Ces interrogations sont nécessaires, mais elles ne doivent pas rester suspendues dans le texte. Chacune doit être transformée en une affirmation localisée, puis reliée à un protocole définissant l’objet éprouvé, la portée de la proposition, les conditions expérimentales, les mesures effectuées, les traces conservées et les résultats susceptibles d’en modifier le statut. La structure méthodologique du chapitre peut ainsi être résumée par la chaîne suivante :

où :

- (\mathcal C) désigne une affirmation scientifiquement délimitée ;
- (\mathcal P) désigne le protocole contractuel par lequel elle est

éprouvée ;

- (\mathcal D_{\mathrm{preuve}}) désigne le dossier de preuve produit

par cette épreuve ;

- (\operatorname{Stat}(\mathcal C)) désigne le statut révisable attribué

à l'affirmation au regard des éléments disponibles. Cette chaîne exprime la thèse centrale du chapitre : un protocole ne valide pas une architecture ; il produit des éléments de preuve qui modifient le statut d'une affirmation déterminée. Le présent chapitre ne cherchera donc pas à démontrer par avance que l'ÉTR est supérieure aux architectures existantes. Il établira les conditions dans lesquelles une telle affirmation — ou toute autre affirmation portant sur l'ÉTR — pourrait être scientifiquement évaluée. Cette posture produit une conséquence plus générale. Les principes formulés ici ne sont pas exclusivement applicables à l'ÉTR. Ils peuvent également servir à examiner un Transformer, un réseau convolutif, un réseau de neurones sur graphes, un moteur documentaire, un système multi-agent ou toute autre architecture computationnelle. Le chapitre atteint ainsi la frontière entre deux entreprises : le traité particulier de l'ÉTR et une théorie générale de l'évaluation des architectures computationnelles. Cette frontière doit être rendue visible, mais non entièrement franchie. L'objet demeure ici la validation de l'ÉTR.

20.1. Une architecture ne se valide pas en bloc

Une architecture contient plusieurs catégories d'affirmations. Certaines portent sur l'existence, la définition ou la distinction d'un objet. D'autres décrivent une propriété mathématique, une relation causale, une capacité algorithmique, une conformité logicielle, une performance expérimentale, une aptitude applicative ou une condition de qualification. Ces affirmations n'ont pas le même statut. Elles ne peuvent donc pas être établies par les mêmes moyens. Une preuve mathématique peut montrer qu'une transformation conserve un invariant sous des hypothèses déterminées. Elle ne démontre pas que le programme censé appliquer cette transformation est exempt d'erreur. Une suite de tests peut établir qu'un programme respecte sa spécification sur un ensemble de situations. Elle ne démontre pas que cette spécification décrit correctement le problème réel auquel le système est destiné.

Une expérience favorable peut montrer une amélioration sur un jeu de données. Elle ne démontre pas que cette amélioration provient du mécanisme théorique avancé. Une réplication indépendante peut renforcer la confiance dans un phénomène empirique. Elle ne transforme pas ce phénomène en théorème. Une conformité réglementaire peut autoriser ou encadrer un usage. Elle ne démontre ni la cohérence scientifique de l'architecture ni la justesse de ses résultats. Il faut donc renoncer à la formulation générale :

é

et lui substituer des propositions localisées :

é é é è é

Une validation ne porte jamais sur l'architecture entière indépendamment de ses conditions. Elle porte sur une affirmation déterminée, concernant une définition, une famille algorithmique, une implémentation, une version ou une expérience déterminée, dans un domaine et une portée explicitement déclarés. L'ÉTR doit donc être considérée non comme un bloc à défendre ou à rejeter, mais comme un système d'affirmations localisées, interdépendantes et révisables. Une contradiction affectant une affirmation ne réfute pas nécessairement l'ensemble du système. Inversement, la confirmation d'une propriété particulière ne valide pas automatiquement toutes les autres. La première fonction d'une architecture scientifique de la preuve consiste précisément à déterminer ce qui est atteint lorsqu'un résultat change.

———

20.2. La grammaire normative de l'affirmation scientifique

Toute affirmation portant sur une architecture doit pouvoir être représentée par une structure minimale :

où :

- (O) désigne l'objet étudié ;
- (P) désigne la propriété attribuée à cet objet ;
- (\Sigma) désigne la portée scientifique de l'affirmation ;
- (H) désigne les hypothèses nécessaires ;
- (\mu) désigne la méthode générale d'épreuve ;
- (A) désigne les critères permettant de soutenir provisoirement

l'affirmation ;

- (F) désigne les conditions défavorables susceptibles de la réfuter, de

la limiter ou d'imposer sa révision. Cette écriture n'est pas une notation décorative. Elle constitue une discipline de formulation. Dire que « l'ÉTR est plus robuste » ne précise ni l'objet concerné, ni le type de perturbation auquel la robustesse se rapporte, ni la métrique retenue, ni le système de comparaison, ni l'amplitude de variation considérée comme acceptable. Une affirmation exploitable serait plus précisément formulée ainsi : Dans un environnement où les données, le référentiel, les paramètres, la version logicielle et les sources d'aléa restent constants, deux exécutions déterministes de la même chaîne de qualification doivent produire une structure relationnelle identique, à la tolérance numérique préalablement déclarée. L'objet est identifié. La propriété est localisée. Les hypothèses sont explicites. La méthode d'épreuve peut être construite. Un résultat incompatible peut être reconnu. L'affirmation entre alors dans le domaine de l'épreuve scientifique. La grammaire impose également de distinguer plusieurs propositions souvent confondues :

é é é é é é é é é é

Ces affirmations peuvent être liées, mais elles ne sont pas interchangeables.

Une démonstration de (\mathcal C_1) ne suffit pas à établir (\mathcal C_3). Une observation compatible avec (\mathcal C_4) ne suffit pas à établir (\mathcal C_1) pour toutes les entrées admissibles.

Encadré — Les deux grammaires de la preuve

La grammaire (\mathcal C) décrit ce qui est affirmé.
La grammaire (\mathcal P), définie plus loin, décrit comment cette affirmation est éprouvée. Leur séparation empêche qu'une expérience soit confondue avec la proposition qu'elle examine. Une même affirmation peut être soumise à plusieurs protocoles indépendants. Un même protocole peut également produire des éléments de preuve concernant plusieurs affirmations distinctes.

———

20.3. La portée scientifique d'une affirmation

Le domaine d'une affirmation ne suffit pas à en déterminer la signification. Deux propositions peuvent concerner le même domaine tout en possédant des extensions scientifiques très différentes. Considérons l'énoncé :

Cette proposition peut signifier : . que l'ordre est conservé par la définition abstraite de Compose ; . qu'il est conservé par tout algorithme conforme à cette définition ; . qu'il est conservé par une implémentation particulière ; . qu'il a été conservé dans une version déterminée du logiciel ; . qu'il a été conservé dans les expériences effectivement réalisées ; . qu'il est conservé seulement pour certaines catégories d'entrées ; . qu'il est conservé exactement ou seulement à une tolérance donnée. Ces formulations ne sont pas équivalentes. La portée scientifique peut être représentée par :

où :

- (D) désigne le domaine étudié ;
- (\mathscr P) désigne la portée ou l'étendue des objets concernés ;
- (G) désigne le degré de généralité revendiqué ;
- (\varepsilon) désigne la précision, la tolérance ou le régime

d'approximation associé à la proposition.

La portée (\mathscr P) peut notamment distinguer :

é é é

Le degré de généralité (G) indique si la proposition vaut :

- pour toutes les entrées admissibles ;
- pour une classe déterminée d'entrées ;
- pour une distribution donnée ;
- pour un échantillon observé ;
- pour un ensemble de cas spécifiés ;
- pour une situation unique.

La précision (\varepsilon) indique si la propriété est :

- exacte ;
- approximative ;
- numérique ;
- probabiliste ;
- statistique ;
- conditionnelle ;
- valable sous une marge d'erreur déclarée.

Une affirmation ne doit jamais être étendue au-delà de la portée réellement examinée.

Un résultat obtenu sur une implémentation ne démontre pas automatiquement une propriété de toutes les implémentations possibles. Une expérience conduite sur un domaine restreint ne démontre pas une propriété universelle. Une stabilité observée à une certaine tolérance ne doit pas être transformée en égalité exacte. La portée n'est donc pas une réserve ajoutée après la conclusion. Elle constitue une partie de la conclusion elle-même. Une conclusion scientifique complète ne dit pas seulement ce qui est soutenu. Elle dit jusqu'où cela l'est.

20.4. Les dimensions du statut scientifique

La validation ne doit pas être réduite à une opposition entre trois états :

é é é é é

La pratique scientifique produit des statuts plus nuancés. Mais ces statuts ne doivent pas davantage être disposés sur une échelle unique. Une propriété formellement démontrée et un phénomène empiriquement répliqué ne relèvent pas du même axe. Une démonstration mathématique peut être complète sans qu'aucune implémentation n'ait encore été testée. À l'inverse, un phénomène peut être empiriquement robuste sans disposer d'une démonstration formelle générale. Le statut d'une affirmation doit donc être décrit par plusieurs dimensions. On pourra écrire :

où :

- (\operatorname{Stat}_{\mathrm{formel}}) décrit l'état de justification

conceptuelle ou mathématique ;

- (\operatorname{Stat}_{\mathrm{impl}}) décrit la conformité

algorithmique et logicielle ;

- (\operatorname{Stat}_{\mathrm{emp}}) décrit l'état de soutien

expérimental ;

- (\operatorname{Stat}_{\mathrm{usage}}) décrit l'état de validation

applicative et de qualification.

20.4.1. Statut formel

Le statut formel peut être décrit par :

É

avec :

- (\mathrm{NF}) : non formalisé ;
- (\mathrm{Conj}) : conjecturé ou formulé sans preuve suffisante ;
- (\mathrm{Part}) : partiellement établi sous certaines hypothèses ;
- (\mathrm{Établ}) : établi formellement dans la portée déclarée.

20.4.2. Statut d'implémentation

Le statut d'implémentation peut être décrit par :

é

avec :

- (\mathrm{NI}) : non implémenté ;
- (\mathrm{Proto}) : implémentation expérimentale ou partielle ;
- (\mathrm{Conf}) : conformité soutenue par des tests ;
- (\mathrm{Vér}) : conformité vérifiée dans la portée et selon les

moyens déclarés.

20.4.3. Statut empirique

Le statut empirique peut être décrit par :

avec :

Symbole	Statut	Signification
(\mathrm H)	Hypothèse empirique	La proposition est formulée, mais ne dispose pas encore d'une épreuve suffisante.
(\mathrm O)	Observation	Le phénomène a été constaté dans une situation déterminée.
(\mathrm S)	Résultat soutenu	Un protocole défini produit des observations compatibles avec l'affirmation.
(\mathrm{Re})	Résultat reproduit	Le résultat est retrouvé avec le même protocole et des conditions équivalentes.
(\mathrm{Rp})	Résultat répliqué	Un résultat compatible est obtenu indépendamment, avec d'autres données, une autre équipe ou une autre implémentation.

20.4.4. Statut d'usage

Le statut d'usage peut enfin distinguer :

avec :

- (\mathrm{NE}) : non évalué en usage ;
- (\mathrm{Pilote}) : expérimenté dans un environnement limité ;
- (\mathrm{ValApp}) : validé pour une application et des conditions

déterminées ;

- (\mathrm{Qual}) : qualifié pour un usage explicitement circonscrit.

Ces dimensions ne forment pas une progression automatique. Une propriété peut être formellement établie, correctement implémentée, empiriquement soutenue, mais non encore répliquée. Elle peut être techniquement robuste sans être qualifiée pour un usage sensible. Elle peut être applicativement utile alors que son explication théorique demeure partielle. Le terme « établi » ne signifie jamais « vrai sans condition ». Il signifie : suffisamment établi pour la portée, les hypothèses, la méthode et le niveau de précision déclarés. Le statut scientifique demeure révisable. Un nouveau résultat peut augmenter ou diminuer le niveau de confiance associé à une affirmation. Il peut également modifier une seule dimension de son statut sans affecter les autres. La confiance scientifique n'est donc pas un label irréversible. Elle est l'état provisoire d'un dossier de preuve.

20.5. La grammaire du protocole

À la grammaire de l'affirmation répond une grammaire du protocole :

où :

- (I) désigne les entrées et leur provenance ;
- (V) désigne la version exacte du système examiné ;
- (C) désigne les conditions contrôlées ;
- (\mathscr E) désigne la procédure expérimentale ;
- (B) désigne les systèmes de référence, les témoins ou les conditions

de comparaison ;

- (\mathcal M) désigne l'ensemble des métriques utilisées ;
- (\tau) désigne les seuils d'interprétation ;
- (F) désigne les conditions d'échec, de réfutation, de limitation ou

d'inconclusion ;

- (T) désigne les traces nécessaires à la reproduction, à l'audit et à la

contestation. Une expérience n'est donc pas définie uniquement par les données qu'elle utilise et le score qu'elle produit. Elle doit indiquer quelle version de l'architecture a été examinée, quelles ressources lui ont été attribuées, quelles variations ont été autorisées, quelles configurations ont servi de contrôle, quelles mesures ont été retenues et comment un résultat contraire sera interprété. Les critères d'interprétation doivent être déterminés avant l'analyse finale. Une métrique ajoutée après l'observation des résultats, un seuil déplacé afin de préserver une hypothèse ou une exclusion décidée parce qu'un cas contredit la tendance initiale ne possèdent pas le même statut qu'une procédure préalablement définie. Toute modification du protocole demeure possible. Elle doit cependant être enregistrée comme une déviation, justifiée et distinguée du protocole initial. Le protocole doit également préciser ce qu'il n'est pas capable d'établir. Cette exigence est essentielle. Une expérience consacrée à la stabilité numérique ne peut pas, à elle seule, conclure sur la pertinence sémantique du système. Un protocole de performance applicative ne démontre pas automatiquement l'existence du mécanisme causal revendiqué. La validité d'un protocole dépend donc autant de la précision de ses conclusions positives que de la clarté de ses limites.

20.6. Le protocole comme objet scientifique : le contrat expérimental

Un protocole ne doit pas seulement décrire une expérience. Il doit posséder son propre contrat. Cette exigence prolonge les contrats déjà attribués aux objets de l'ÉTR. Si Compose, la qualification ou le routeur déclarent leurs entrées, leurs préconditions et leurs sorties, le protocole chargé de les examiner doit être soumis à une discipline comparable. Le contrat expérimental peut être représenté par :

où :

- (I) désigne les entrées admises ;
- (Q) désigne les préconditions nécessaires ;
- (J) désigne les invariants du protocole ;
- (\mathscr E) désigne les opérations expérimentales ;
- (O) désigne les sorties produites ;
- (\mathcal M) désigne les mesures applicables ;
- (F) désigne les conditions d'échec ou d'invalidité ;
- (T) désigne les exigences de traçabilité.

Les préconditions (Q) déterminent les situations dans lesquelles le protocole possède une signification. Un protocole destiné à examiner un comportement déterministe ne peut pas être appliqué sans adaptation à un système dont les sources d'aléa ne sont ni identifiées ni contrôlées. Un protocole consacré à l'effet d'un référentiel ne peut pas être interprété si plusieurs autres variables changent simultanément sans avoir été enregistrées. Les invariants (J) définissent ce qui

doit demeurer constant pendant l'expérience : version du code, budget, référentiel, corpus, ordre des opérations, politique de décision, matériel, méthode de prétraitement ou

procédure d'échantillonnage. Les sorties (O) ne se réduisent pas à un score. Elles peuvent inclure :

- des journaux d'exécution ;
- des états intermédiaires ;
- des structures relationnelles ;
- des matrices de confusion ;
- des distributions d'erreurs ;
- des traces de routage ;
- des abstentions ;
- des preuves de provenance ;
- des événements de dégradation ;
- des résultats négatifs ;
- des cas non interprétables.

Le contrat expérimental permet ainsi de contrôler le protocole lui-même. Un résultat ne peut être interprété lorsque les préconditions du protocole ne sont pas satisfaites. Une expérience ne peut être déclarée comparable lorsque ses invariants ont été modifiés sans justification. Une conclusion ne peut être reproduite lorsque ses sorties et ses traces n'ont pas été conservées. Le protocole devient alors un objet scientifique autonome : définissable, spécifiable, versionnable, testable, auditable et révisable.

20.7. L'espace expérimental

Une expérience n'existe jamais isolément. Elle appartient à un espace expérimental. Cet espace peut être représenté par :

où :

- (\mathcal X) désigne l'espace des entrées ;
- (\Theta) désigne l'espace des paramètres ;
- (\mathcal R) désigne l'ensemble des référentiels applicables ;
- (\mathcal C) désigne les contraintes matérielles, temporelles,

humaines et opérationnelles ;

- (\mathcal B) désigne le budget attribué à l'expérience.

Le budget (\mathcal B) peut inclure :

- le temps de calcul ;
- la mémoire ;
- l'énergie ;
- le volume de données ;
- le nombre d'essais de réglage ;
- l'intervention humaine ;
- le coût financier ;
- les ressources d'annotation ;
- les ressources de supervision ;
- les coûts d'instrumentation et d'audit.

Deux expériences ne sont pas comparables simplement parce qu'elles produisent une métrique portant le même nom. Elles doivent appartenir à des espaces identiques ou suffisamment compatibles dans les dimensions qui affectent l'affirmation étudiée. On pourra écrire :

pour signifier qu'une transformation déclarée (\Phi) rend les deux espaces

comparables relativement à l'ensemble de contraintes pertinentes

(\mathcal C_*).

La relation d'équivalence est donc relative à la question examinée. Deux expériences peuvent être comparables pour la latence sans l'être pour l'énergie. Elles peuvent être comparables pour l'exactitude sans l'être pour l'intervention humaine. Elles peuvent être comparables pour une fonction de recherche sans l'être pour l'auditabilité. Cette compatibilité ne requiert pas toujours une identité absolue. Deux systèmes peuvent utiliser des matériels différents si leur consommation ou leur débit sont normalisés selon une méthode justifiée. Deux expériences peuvent employer des jeux de données distincts si ceux-ci représentent des distributions comparables selon des critères contrôlés. Deux architectures peuvent posséder des nombres de paramètres différents si la comparaison porte explicitement sur une contrainte de

latence, d'énergie ou de coût plutôt que sur leur taille. L'essentiel est que l'équivalence retenue soit déclarée avant l'interprétation et limitée aux dimensions pour lesquelles elle est défendable. Sans définition de l'espace expérimental, une comparaison risque de confondre les propriétés du système avec celles de son environnement.

20.8. La chaîne de validité

La validation de l'ÉTR doit être organisée selon plusieurs niveaux liés sans être confondus :

é

La validation conceptuelle établit que les objets sont définis, distincts et compatibles. La validation formelle examine les équations, les transformations, les propriétés, les domaines et les invariants. La validation algorithmique détermine si les procédures proposées réalisent effectivement les opérations formelles. La validation logicielle vérifie si le programme implémente fidèlement ces procédures dans les conditions déclarées. La validation expérimentale observe le comportement du système dans des environnements contrôlés. La validation applicative mesure son aptitude à résoudre un problème réel. La qualification examine enfin les conditions dans lesquelles ce système

peut être utilisé : exigences juridiques, réglementaires, sectorielles, éthiques, organisationnelles et opérationnelles. Ces niveaux forment une chaîne de dépendances, mais non une simple succession chronologique. Si la définition de la qualification demeure ambiguë, il devient impossible de déterminer ce qu'une expérience consacrée à cette fonction mesure réellement. Si Compose est formellement défini mais que le programme lui ajoute silencieusement une règle, les résultats du logiciel ne peuvent plus être attribués à Compose tel qu'il a été théorisé. Si une performance est observée sur des données déjà utilisées pour construire ou régler le système, elle ne permet pas d'établir sa capacité de généralisation. Si un prototype fonctionne dans un environnement contrôlé, cela ne suffit pas à le qualifier pour un environnement critique. Une faiblesse à un niveau ne rend pas nécessairement tout le système inutile. Elle limite toutefois la portée des conclusions pouvant être tirées aux niveaux qui en dépendent. La chaîne de validité protège ainsi contre deux erreurs opposées :

- attribuer à la théorie les défaillances propres à une implémentation ;
- attribuer indéfiniment au logiciel les contradictions qui atteignent en

réalité la théorie.

20.9. Les invariants scientifiques

Une expérience ne mesure pas seulement ce qui varie. Elle doit également établir ce qui n'a pas changé. Cette seconde dimension est fondamentale. Une architecture peut augmenter son exactitude tout en détruisant une propriété qu'elle avait précisément pour fonction de conserver. Elle peut réduire sa latence en supprimant la provenance des transformations. Elle peut améliorer un score global en fusionnant la relation et la décision. Elle peut produire une représentation plus compacte en rendant impossible la reconstruction des étapes antérieures. Une amélioration de performance peut donc constituer une régression scientifique. On appellera invariant scientifique toute propriété qui doit demeurer stable au cours d'une transformation, d'une composition, d'une optimisation ou d'une évolution de l'architecture, sauf déclaration explicite du contraire.

Pour une transformation :

et un invariant (J), la propriété de conservation peut s'écrire :

Cette égalité peut être exacte, approximative, probabiliste ou conditionnelle. Son statut doit être précisé. Dans l'ÉTR, les invariants possibles dépendent des objets étudiés. Pour Compose, l'ordre des éléments, leur typage, leur identité ou leur provenance peuvent constituer des invariants lorsque la spécification exige leur conservation. Pour le routeur, l'invariant peut porter sur la politique de routage : à conditions identiques, un changement extérieur à cette politique ne doit pas modifier arbitrairement la destination. Pour la qualification, le référentiel utilisé doit rester identifiable. Une qualification dont le résultat subsiste mais dont le référentiel disparaît ne conserve pas entièrement sa signification. Pour la mémoire relationnelle, la provenance, la temporalité, l'identité des objets et la nature des relations peuvent constituer des invariants plus importants que la simple présence d'une information. La validation doit donc accompagner chaque mesure de performance d'un contrôle des invariants concernés. Une transformation peut être décrite par son bilan structurel :

où :

- (K) représente les propriétés conservées ;
- (X_f) représente les propriétés transformées ;
- (A) représente les propriétés ajoutées ;
- (S) représente les propriétés supprimées ;
- (U) représente les propriétés dont le statut reste indéterminé.

Cette dernière catégorie est essentielle.

Une propriété ne doit pas être considérée comme conservée simplement parce que sa perte n'a pas été mesurée.

é

20.10. Variables, dépendances et observabilité

Une validation rigoureuse exige de distinguer les catégories de variables mobilisées par l'architecture et par le protocole. La variable indépendante est manipulée par le protocole. Elle constitue la cause expérimentale dont on cherche à mesurer les effets : changement de référentiel, suppression de Compose, variation du volume de mémoire, modification de la politique de routage ou altération contrôlée des relations. La variable dépendante est la grandeur dont on observe la variation : exactitude, stabilité, coût, nombre d'erreurs, temps de restitution, conservation d'un invariant ou taux d'abstention. La variable contrôlée est maintenue constante afin d'empêcher qu'elle explique à tort la variation observée. La variable observée est directement accessible à la

mesure : latence, destination choisie, relation enregistrée, valeur restituée ou quantité de mémoire utilisée. La variable calculée résulte d'une opération déterminée sur plusieurs observations : moyenne, variance, score de stabilité, taux d'erreur ou coût agrégé. La variable dérivée combine plusieurs mesures afin de représenter une propriété plus générale. Un indice d'auditabilité peut, par exemple, agréger le taux de reconstruction, la présence de la provenance et la localisation des transformations. La variable latente désigne une propriété supposée mais non directement observable. Elle ne peut être admise sans indicateurs, sans modèle d'inférence ou sans expérience permettant d'en isoler les effets. La variable confondante influence à la fois la variable indépendante et la variable dépendante. Elle peut produire une relation apparente qui n'est pas causale. La variable médiatrice décrit un mécanisme par lequel l'intervention produit son effet.

Cette classification prévient une confusion fréquente entre ce qui est mesuré et ce qui est interprété. Lorsqu'un système produit un score élevé de cohérence, ce score est une variable observée ou calculée. La « compréhension » qu'on pourrait lui attribuer demeure une interprétation, à moins qu'un protocole indépendant n'en définisse les manifestations, les limites et les conditions défavorables. La même exigence s'applique à toutes les architectures. Dans un Transformer, les poids d'attention peuvent être observés, mais leur signification causale ne doit pas être déduite de leur seule présence. Dans un réseau convolutif, l'activation d'un filtre peut être mesurée, mais elle ne constitue pas automatiquement une explication. Dans l'ÉTR, une relation enregistrée peut être observée, mais son adéquation au phénomène étudié doit encore être validée. La classification des variables oblige ainsi à séparer :

é

———

20.11. Le socle fonctionnel commun des architectures

Toute architecture computationnelle peut être ramenée à une structure minimale :

où une entrée (X) est transformée par une opération (T) en une sortie (Y). À ce niveau d'abstraction, les architectures partagent plusieurs éléments : une entrée, une représentation, une transformation, un état intermédiaire éventuel, une sortie et un critère d'évaluation. Leur différence apparaît dans les objets qu'elles rendent explicites et dans les opérations qu'elles privilégient. Un réseau convolutif introduit la convolution locale et hiérarchique comme principe de transformation.

Un Transformer introduit l'attention comme mécanisme de mise en relation contextuelle. Un réseau de neurones sur graphes organise la propagation à travers des nœuds, des arêtes et des règles d'agrégation. L'ÉTR rend notamment explicites le référentiel, la qualification, la relation, la mémoire relationnelle et le routage. Ces architectures ne doivent donc pas être décrites uniquement par leur nom ou leur famille historique. Elles peuvent être examinées à partir des fonctions qu'elles réalisent :

Aucune architecture n'est tenue de réaliser l'ensemble de ces fonctions. Une même fonction peut également être répartie entre plusieurs composants. Cette description fonctionnelle permet une comparaison plus équitable. Une fonction de recherche doit être comparée à une autre fonction de recherche. Une mémoire relationnelle doit être confrontée à un autre mécanisme de mémoire. Une politique de routage doit être examinée face à une politique remplissant un rôle comparable. Comparer des architectures uniquement comme des blocs indivisibles revient souvent à confronter des systèmes qui ne répondent ni aux mêmes problèmes ni aux mêmes contraintes. Comparer leurs fonctions permet au contraire de déterminer leurs complémentarités, leurs recouvrements, leurs différences et leurs éventuels déplacements d'ontologie.

20.12. Optimisation, extension et rupture scientifique

Toutes les différences entre architectures ne constituent pas des ruptures. Une optimisation améliore une propriété mesurable sans modifier les objets fondamentaux du système. Réduire la latence d'une opération, améliorer la précision d'un modèle ou diminuer sa consommation mémoire relève généralement de cette catégorie. Une extension architecturale ajoute une fonction, un composant ou un mécanisme tout en conservant l'ontologie principale du système. Une rupture scientifique apparaît lorsqu'un objet auparavant implicite devient explicite, lorsqu'une transformation jusque-là confondue avec une autre reçoit une autonomie, ou lorsqu'un nouveau niveau d'organisation modifie la manière même de formuler les problèmes. La rupture ne réside donc pas nécessairement dans une amélioration immédiate des performances. Elle peut résider dans la possibilité :

- de poser de nouvelles questions ;
- de séparer des causes auparavant fusionnées ;
- d'intervenir sur un objet sans modifier les autres ;
- de construire des protocoles jusque-là impossibles ;
- de conserver de nouveaux invariants ;
- de rendre certaines décisions traçables ou contestables.

Plusieurs déplacements proposés par l'ÉTR peuvent être examinés sous cet angle :

é é é é éé é é é é é é

Ces déplacements ne doivent pas être déclarés révolutionnaires par définition. Leur portée doit être éprouvée. Pour chacun, le protocole doit déterminer si l'objet nouvellement explicité permet :

- de mesurer une propriété auparavant inaccessible ;
- de localiser une erreur qui demeurait indifférenciée ;
- de réaliser une intervention expérimentale indépendante ;

- d’améliorer la traçabilité ;
- de réduire le coût de preuve ;
- de produire une capacité opérationnelle nouvelle ;
- d’obtenir ces avantages sans coût disproportionné.

Une rupture scientifique n’est pas proclamée par le vocabulaire qui l’accompagne. Elle est établie lorsque la nouvelle décomposition produit des capacités d’analyse, de contrôle ou d’action qui ne pouvaient pas être obtenues de manière équivalente dans le cadre précédent.

20.13. Le graphe des protocoles

Les protocoles de validation ne sont pas indépendants. Ils forment un système de dépendances dans lequel les conclusions de certains protocoles deviennent les préconditions d’autres protocoles. Ce système peut être représenté par un graphe :

où :

- (V_{\mathcal P}) représente les protocoles ;
- (E_{\mathcal P}) représente leurs relations de dépendance.

Pour l’ÉTR, une structure minimale peut être exprimée ainsi :

é é é

Cette représentation ne signifie pas qu’un protocole antérieur doive être définitivement achevé avant toute expérience ultérieure. Elle signifie que la portée des conclusions ultérieures dépend du niveau de confiance accordé aux résultats antérieurs. La validation du routeur dépend, par exemple, de la stabilité de la qualification et de la relation qu’il exploite. Si ces objets sont mal définis ou instables, une erreur de routage ne peut pas être attribuée avec certitude au routeur lui-même. Le graphe permet également de localiser les conséquences d’une révision. Lorsqu’une propriété de Compose est modifiée, tous les protocoles dépendant de cette propriété doivent être identifiés. Certaines expériences devront être répétées. D’autres resteront valides si elles n’utilisaient pas l’invariant ou la propriété révisés. On peut associer à chaque arête du graphe une dépendance explicite :

pour signifier que le protocole (\mathcal P_j) dépend d’une affirmation (\mathcal C_k) soutenue ou établie par (\mathcal P_i).

Cette organisation transforme la validation en infrastructure cumulative plutôt qu’en succession d’expériences isolées.

20.14. Protocole de validation conceptuelle

La première validation de l'ÉTR porte sur l'identité de ses objets. Chaque terme fondamental doit disposer :

- d'une définition normative ;
- d'un domaine d'emploi ;
- d'un contrat fonctionnel ;
- de critères d'identité ;
- de critères de distinction ;
- d'exclusions suffisamment précises pour empêcher sa fusion avec les

objets voisins. Le protocole conceptuel peut être mené par reconstruction indépendante. Plusieurs lecteurs ou évaluateurs reçoivent les définitions, les exemples et les contrats d'entrée et de sortie. Ils doivent pouvoir : . identifier les objets mobilisés dans une chaîne donnée ; . distinguer les opérations successives ; . reconstruire l'ordre des transformations ; . localiser les dépendances ; . reconnaître les cas où un terme est utilisé hors de son domaine ; . résoudre les désaccords en revenant à une source normative. Le protocole réussit lorsque les divergences d'interprétation peuvent être résolues par référence à une définition localisée. Il échoue notamment lorsqu'un même terme désigne plusieurs fonctions incompatibles, lorsqu'une opération dépend d'un objet non défini, lorsqu'une distinction annoncée disparaît dans les exemples ou lorsqu'aucune source normative ne permet de trancher les divergences. La validation conceptuelle ne requiert pas une identité absolue de formulation entre tous les lecteurs. Elle exige que l'architecture possède une structure normative suffisamment précise pour que les désaccords puissent être localisés et résolus.

20.15. Protocole de validation formelle

La validation formelle examine les propriétés des objets et des transformations. Pour chaque opération, le protocole doit préciser :

- son domaine d'entrée ;
- son domaine de sortie ;
- ses préconditions ;
- ses invariants ;
- ses effets ;
- ses cas limites ;
- ses conditions d'échec ;
- ses propriétés algébriques éventuelles ;

- ses relations avec les autres opérations.

Les propriétés générales doivent être démontrées lorsqu'une démonstration est possible. Les propriétés portant sur des espaces finis peuvent être vérifiées exhaustivement. Les propriétés calculatoires peuvent être soumises à des tests génératifs, à des tests fondés sur les invariants ou à des méthodes de vérification formelle. Aucune propriété algébrique ne doit être présumée. L'associativité, la commutativité, l'idempotence, l'existence d'un élément neutre, l'inversibilité ou la réversibilité de Compose ne doivent être affirmées que si elles appartiennent effectivement à sa définition et si elles ont été établies dans une portée explicite. La notation mathématique ne remplace pas la preuve. Elle permet seulement de formuler avec précision ce qui doit être prouvé, réfuté ou laissé indéterminé. La validation formelle doit aussi distinguer :

é é é é é é é é é

Une propriété définitoire résulte d'une convention de construction. Une propriété démontrée résulte d'un raisonnement valide sous certaines hypothèses.

Une propriété observée résulte d'une exécution ou d'une mesure. Ces statuts peuvent converger, mais ils ne doivent jamais être confondus.

20.16. Protocole de conformité algorithmique et logicielle

Entre la théorie et l'exécution se trouvent deux traductions :

é é

Une divergence peut apparaître à chaque passage. Toute fonction critique doit donc être rattachée à la propriété qu'elle est censée implémenter. Les entrées, les sorties, les effets de bord, les tolérances numériques, les états internes et les erreurs possibles doivent être déclarés. Une implémentation de référence, volontairement simple, peut servir d'oracle à une version optimisée. Les deux sont exécutées sur les mêmes entrées et leurs résultats sont comparés selon une relation de conformité définie à l'avance. Les tests unitaires vérifient des cas déterminés. Les tests de propriétés examinent les invariants sur des ensembles d'entrées générées. Les tests différentiels comparent plusieurs implémentations indépendantes. Les tests de mutation contrôlent la capacité de la suite de tests à détecter des altérations volontaires. Les tests d'intégration examinent les effets produits lors du passage d'un composant au suivant. Les tests de non-régression vérifient qu'une évolution du système ne détruit pas des propriétés antérieurement établies. Les tests de précision numérique examinent les effets de représentation, d'arrondi, d'ordre d'opération et de matériel. Une implémentation n'est pas conforme parce qu'elle ne provoque aucune erreur visible. Elle l'est lorsque les opérations prévues peuvent être localisées dans le code, que leurs sorties correspondent à la spécification et que leurs propriétés résistent aux contrôles définis.

20.17. Protocoles propres aux objets de l’ÉTR

Les protocoles généraux doivent être appliqués aux objets particuliers de l’architecture.

20.17.1. Validation de Compose

Compose doit être évalué selon les propriétés qui lui sont effectivement attribuées. Le protocole spécifie :

- l’ordre de composition ;
- les types acceptés ;
- les préconditions ;
- les propriétés conservées ;
- les transformations autorisées ;
- les informations ajoutées ;
- les informations supprimées ;
- les cas où la composition doit échouer ou être refusée.

Une étude d’ablation compare ensuite plusieurs conditions :

î è é é é é é é é â é é

La comparaison doit mesurer non seulement le résultat final, mais également les invariants concernés. Si la performance augmente alors que l’ordre, le typage ou la provenance sont détruits, l’amélioration doit être rapportée avec cette perte. Elle ne peut pas être présentée comme un progrès global de Compose. La contribution de Compose est soutenue lorsqu’elle est reproductible, localisée, compatible avec le mécanisme annoncé et absente ou diminuée lorsque les propriétés essentielles de Compose sont retirées.

20.17.2. Validation des référentiels

L’explicitation du référentiel implique une double exigence. À référentiel constant, les résultats doivent demeurer stables dans les limites prévues.

Lorsqu’un référentiel est modifié, les variations doivent correspondre aux dépendances déclarées. Le protocole exécute une même entrée sous plusieurs référentiels :

en maintenant constantes les autres variables. Il mesure :

- la stabilité interne pour chaque référentiel ;
- la sensibilité aux variations prévues ;
- les effets non anticipés ;

- la localisation des changements ;
- la conservation des propriétés déclarées indépendantes du

référentiel ;

- la capacité à reconstruire le référentiel ayant produit un résultat.

Le protocole échoue lorsqu'un changement extérieur au référentiel modifie arbitrairement la qualification ou lorsqu'une variation locale du référentiel produit des effets globaux non attribuables.

20.17.3. Validation de la qualification

La qualification doit être distinguée de la représentation initiale et de la décision finale. Le protocole fixe l'entrée et le référentiel, puis examine si la qualification produite correspond aux règles annoncées. Il modifie ensuite le référentiel en maintenant l'entrée constante, puis l'entrée en maintenant le référentiel constant. Cette intervention croisée peut être représentée par :

Elle permet de distinguer l'effet de l'entrée de celui du référentiel. La qualification est soutenue comme transformation autonome lorsque ses variations peuvent être attribuées soit à l'entrée, soit au référentiel, sans dépendre silencieusement de la décision qui sera prise ensuite.

20.17.4. Validation de la séparation entre relation et décision

Cette séparation doit être éprouvée par intervention. Dans une première expérience, la structure relationnelle est maintenue constante tandis que plusieurs politiques de décision sont appliquées :

Les décisions peuvent varier. La relation ne doit cependant pas être réécrite afin de correspondre au choix final. Dans une seconde expérience, la politique de décision reste constante tandis que la relation est modifiée de manière contrôlée :

Les variations de décision doivent alors pouvoir être expliquées par les variations relationnelles introduites. La séparation échoue si la couche de décision altère silencieusement les relations dont elle dépend, si la relation contient déjà une décision irréversible ou si les deux couches ne peuvent pas être observées et modifiées indépendamment. Sans ce protocole, la séparation entre relation et décision resterait une distinction terminologique.

20.17.5. Validation de la mémoire relationnelle

La mémoire doit être évaluée comme structure de conservation, de mise à jour, de contradiction et de restitution. Le protocole introduit des informations dont la provenance, la temporalité et les relations sont connues. Il demande ensuite au système de :

- les restituer ;
- les compléter ;
- les réviser ;
- les relier ;
- distinguer leurs versions ;
- signaler leurs contradictions ;
- identifier leur obsolescence ;
- reconstruire leur provenance.

L'évaluation distingue :

- la présence de l'information ;
- son accessibilité ;
- la fidélité de sa restitution ;
- l'exactitude de sa provenance ;
- la conservation de ses relations ;
- la gestion de sa temporalité ;
- la gestion de son obsolescence ;
- la possibilité de reconstruire son évolution.

Une mémoire qui restitue une valeur correcte tout en lui attribuant une provenance erronée ne satisfait pas entièrement la traçabilité. Une mémoire qui conserve toutes les versions sans distinguer leur validité temporelle ne satisfait pas nécessairement la fonction de mise à jour. La validation porte donc sur la structure conservée autant que sur la réponse produite.

20.17.6. Validation du routeur

Le routeur doit être évalué selon les décisions qu'il est autorisé à prendre. Un corpus de situations est construit avec, pour chacune :

- une destination attendue ;
- un ensemble de destinations admissibles ;
- une condition d'abstention ;
- un coût d'erreur ;
- un degré d'ambiguïté ;

- les informations disponibles au moment de la décision.

Les cas déterminés doivent être distingués des cas ambigus. Le protocole mesure :

- les erreurs de destination ;
- le coût de ces erreurs ;
- les abstentions correctes ;
- les abstentions excessives ;
- les décisions prises avec information insuffisante ;
- la stabilité de la politique de routage ;
- la capacité à expliquer les variables ayant déterminé la destination.

Un taux global ne suffit pas. Une erreur rare mais grave peut être masquée par un grand nombre de décisions triviales. Le routeur doit donc être évalué par type de situation, par coût de mauvaise orientation et par niveau d'incertitude.

20.18. Études d'ablation, contrôles négatifs et contrefactuels

Une performance globale ne démontre pas la cause de cette performance. L'étude d'ablation retire ou neutralise successivement les composants de l'architecture. Elle examine ce qui se produit lorsque :

- Compose disparaît ;
- le référentiel n'est plus explicite ;
- la mémoire relationnelle est remplacée par une conservation simple ;
- la décision accède directement aux données sans passer par la

relation ;

- le routeur est remplacé par une règle fixe ;
- la provenance est supprimée ;
- l'ordre est aléatoirement perturbé.

Une ablation doit conserver le reste du système aussi constant que possible. Elle ne doit pas comparer une architecture complète et optimisée à un témoin volontairement dégradé par plusieurs modifications simultanées. Les contrôles négatifs vérifient que le système échoue ou se dégrade lorsqu'il ne dispose plus de l'information censée produire sa réussite. Si une architecture conserve sa performance après permutation aléatoire des relations déclarées essentielles, il devient difficile d'affirmer qu'elle utilisait effectivement ces relations. Les expériences contrefactuelles modifient une variable tout en maintenant les autres constantes. Elles répondent à une question causale :

é é é é é

é

Ces méthodes permettent de passer de l'observation d'une performance à l'examen du mécanisme qui la produit. Elles ne suffisent toutefois pas toujours à établir une causalité complète. Une ablation peut modifier plusieurs mécanismes secondaires. Un contrôle négatif peut introduire une perturbation irréaliste. Un contrefactuel peut dépendre d'hypothèses fortes. Le protocole doit donc préciser la portée causale exacte de chaque intervention.

———

20.19. Comparaison expérimentale et coût de preuve

La comparaison doit commencer par l'identification de la fonction commune. Une architecture ne doit pas être confrontée globalement à une autre si leurs objectifs, leurs entrées, leurs sorties ou leurs contraintes ne sont pas

comparables. Plusieurs régimes d'équité peuvent être retenus :

- volume de données constant ;
- temps de calcul constant ;
- latence maximale constante ;
- consommation mémoire constante ;
- budget énergétique constant ;
- coût financier constant ;
- nombre de paramètres constant ;
- intervention humaine constante ;
- coût total de déploiement constant.

Aucun de ces régimes n'est universel. Le protocole doit indiquer celui qui est utilisé et les conclusions qu'il autorise. Une comparaison complète sépare au moins deux catégories de résultats. La première concerne la résolution de la tâche :

- exactitude ;
- précision ;
- rappel ;
- erreur ;
- latence ;
- débit ;
- coût ;
- taux d'abstention ;
- robustesse.

La seconde concerne les propriétés architecturales :

- traçabilité ;
- conservation des invariants ;
- auditabilité ;
- composabilité ;
- stabilité ;
- coût de mise à jour ;
- capacité de contestation ;
- localisation des erreurs ;
- réversibilité des transformations.

Pour une métrique (m), l'écart entre l'ÉTR et une référence peut s'écrire :

É é é

Cette différence doit être accompagnée de son incertitude, de sa

dispersion, de la taille de l'effet et du régime de comparaison utilisé. Mais le coût calculatoire ne constitue qu'une partie du coût total. Une architecture possède également un coût de preuve : l'ensemble des ressources nécessaires pour comprendre, spécifier, vérifier, auditer, reproduire et contester une affirmation. Ce coût doit être rapporté à une affirmation, à un protocole et à un espace expérimental déterminés :

Il peut être décomposé ainsi :

é é ç é

Le coût de spécification correspond au travail nécessaire pour définir précisément l'objet, la propriété et la portée étudiés. Le coût d'instrumentation correspond aux mécanismes requis pour observer les états internes et produire les mesures nécessaires. Le coût d'exécution comprend les ressources mobilisées par les expériences. Le coût de traçage correspond à la conservation de la provenance, des états intermédiaires et des conditions d'exécution.

Le coût d'audit représente l'effort nécessaire pour reconstruire et contrôler le résultat. Le coût de réplication désigne les ressources requises pour produire une preuve indépendante. Ce coût varie selon l'affirmation. Démontrer un invariant simple, reproduire une expérience de qualification et qualifier un système robotique complet ne demandent pas les mêmes ressources. Une architecture peut être extrêmement performante tout en possédant un coût de preuve élevé si elle exige des centaines de paramètres opaques, des interventions manuelles difficiles à reconstituer ou une infrastructure inaccessible. À l'inverse, une architecture légèrement moins performante peut présenter un coût de preuve inférieur si ses transformations sont localisées, ses invariants contrôlables et ses résultats facilement reproductibles. Le coût de preuve ne remplace pas les mesures de performance. Il complète leur interprétation. Une architecture scientifique n'est pas

seulement un système capable de produire un résultat. C'est également un système dont les résultats peuvent être raisonnablement établis.

20.20. Robustesse et domaine de validité

La robustesse n'existe pas sans perturbation définie. Un système peut être robuste aux erreurs typographiques et sensible aux changements de référentiel. Il peut résister au bruit numérique et échouer devant une contradiction sémantique. Il peut généraliser à des données proches de son domaine initial et devenir instable lors d'un changement de distribution. Le protocole doit donc définir chaque classe de perturbation par :

é é

La nature précise ce qui est modifié. L'amplitude précise l'intensité de la perturbation. La fréquence précise son occurrence ou sa densité. La plausibilité précise si la perturbation correspond à une situation réaliste, extrême ou purement diagnostique. La dégradation attendue doit également être décrite. Un système robuste n'est pas nécessairement un système dont le résultat ne change jamais. Il peut s'agir d'un système :

- dont la performance diminue progressivement ;
- dont l'incertitude augmente correctement ;
- dont les sorties deviennent plus prudentes ;
- qui localise les conditions devenues invalides ;
- qui refuse de conclure lorsque ses hypothèses ne sont plus

satisfaites. Une abstention explicite peut être scientifiquement préférable à une réponse apparemment stable mais dépourvue de fondement. La robustesse doit donc être rapportée à un domaine de validité :

où (N_{\max}) désigne le niveau maximal de perturbation pour lequel la
propriété demeure soutenue dans la tolérance (\varepsilon).

20.21. Typologie des erreurs

Toutes les erreurs ne sont pas de même nature. Leur regroupement dans une mesure unique empêche souvent de localiser la défaillance réelle. Une typologie minimale peut distinguer :

Type d'erreur	Nature	Protocole

principalement

concerné		
Conceptuelle	Objet ambigu, frontière instable, hypothèse implicite	Validation conceptuelle
Formelle ou mathématique	Propriété fausse, démonstration invalide, invariant non conservé	Validation formelle
Algorithmique	Procédure non conforme à l’opération théorique	Conformité algorithmique
Logicielle	Erreur de code, de dépendance, de précision numérique ou d’état d’exécution	Validation logicielle
Expérimentale	Biais de données, contrôle insuffisant, puissance inadéquate, espace incompatible	Contrat expérimental
Interprétative	Conclusion excédant la portée ou confusion entre observation et propriété	Analyse scientifique
Applicative	Inadéquation entre le système et le besoin réel	Validation applicative
Juridique ou qualificative	Usage incompatible avec les obligations applicables	Qualification

Chaque erreur appelle une réponse différente. Une erreur logicielle peut être corrigée sans modifier la théorie. Une erreur conceptuelle peut exiger la révision des définitions et de tous les protocoles qui en dépendent. Une erreur interprétative peut laisser les résultats bruts intacts tout en invalidant la conclusion qui en avait été tirée. Cette typologie empêche deux confusions opposées. La première consiste à réfuter une théorie sur la seule base d'un programme défectueux. La seconde consiste à protéger indéfiniment une théorie en attribuant toute contradiction à une erreur d'implémentation.

La nature de l'erreur doit être déterminée par des protocoles de localisation, et non choisie en fonction de la conclusion que l'on souhaite préserver.

20.22. Falsification et résultats négatifs

Une théorie devient impossible à falsifier lorsqu'aucun résultat ne peut compter contre elle. Si chaque échec est interprété comme une mauvaise implémentation, un mauvais jeu de données, une mauvaise lecture, un protocole inadapté ou une preuve de la profondeur du système, aucune expérience ne peut plus réellement l'atteindre. Chaque affirmation centrale de l'ÉTR doit donc comporter une condition défavorable identifiable. Si la séparation entre relation et décision est annoncée, une expérience montrant que la décision réécrit nécessairement la relation constitue un résultat contraire à cette affirmation. Si la traçabilité est revendiquée, l'existence répétée de transformations impossibles à reconstruire remet cette propriété en cause. Si l'explicitation du référentiel est censée localiser les variations, des effets arbitraires et non attribuables affaiblissent cette hypothèse. Un résultat négatif peut néanmoins provenir de plusieurs niveaux :

é é é é é é é é

Ces issues doivent être distinguées. L'échec d'un programme ne réfute pas automatiquement la théorie qu'il prétend implémenter. Mais protéger systématiquement la théorie en invoquant l'implémentation la rendrait inaccessible à l'épreuve. Le protocole doit donc déterminer à l'avance les contrôles permettant de localiser l'échec.

Un résultat indéterminé ne constitue ni une validation ni une réfutation. Il signifie que l'expérience n'a pas produit suffisamment d'information pour modifier légitimement le statut de l'affirmation. L'indétermination doit être conservée comme résultat scientifique à part entière. Elle ne doit pas être transformée en soutien implicite par défaut.

20.23. Reproductibilité, réplication et dossier de preuve

Une expérience est reproductible lorsqu'un tiers peut retrouver son résultat à partir des mêmes données, du même code et de conditions équivalentes. Elle est répliquée lorsqu'un tiers obtient un

résultat compatible à travers une nouvelle exécution, un autre échantillon, une autre équipe ou une implémentation indépendante. Le dossier de preuve doit conserver :

- la spécification évaluée ;
- l'affirmation exacte examinée ;
- la version du protocole ;
- la version du code ;
- les données ou leur procédure d'obtention ;
- leur provenance ;
- l'environnement logiciel ;
- les dépendances ;
- les paramètres ;
- les graines aléatoires ;
- les caractéristiques matérielles pertinentes ;
- les journaux d'exécution ;
- les résultats bruts ;
- les états intermédiaires ;
- les erreurs et abstentions ;
- les procédures produisant les tableaux et figures ;
- les déviations par rapport au protocole initial.

Toute intervention manuelle susceptible d'influencer le résultat doit être enregistrée. Le dossier de preuve peut être représenté par :

où :

- ($\mathcal C$) désigne l'affirmation examinée ;
- ($\mathcal P$) désigne le protocole exécuté ;
- ($\mathbb E$) désigne l'espace expérimental ;
- ($\mathcal O$) désigne les observations et résultats bruts ;
- ($\mathcal T$) désigne les traces et éléments de provenance ;
- ($\mathcal A$) désigne l'analyse reliant les observations à l'affirmation.

Le dossier de preuve doit relier chaque résultat à l'affirmation qu'il soutient, limite ou contredit. On peut écrire :

Une même expérience peut soutenir une affirmation et en laisser une autre indéterminée. Un même résultat peut augmenter la confiance dans une propriété tout en révélant la perte d'un invariant

distinct. La validation devient alors un système de preuves localisées plutôt qu'un dossier global destiné à défendre l'architecture dans son ensemble.

20.24. Validation applicative et qualification

Une architecture peut satisfaire ses contrats internes sans résoudre correctement le problème auquel elle est destinée. La validation applicative commence donc par définir le besoin indépendamment de la solution. Elle identifie :

- les utilisateurs ;
- les décisions concernées ;
- les contraintes temporelles ;
- les erreurs acceptables ;
- les conséquences d'une défaillance ;
- les solutions déjà disponibles ;
- les conditions de recours ;
- les exigences de compréhension ;
- les coûts de maintenance et de correction.

L'ÉTR doit ensuite être évaluée dans une situation représentative, avec des données et des contraintes proches de l'usage envisagé. La performance technique n'est qu'une dimension. Le protocole peut également mesurer :

- la compréhension des résultats ;
- le temps nécessaire à leur audit ;
- la capacité de correction ;
- la qualité de la provenance ;
- la possibilité de contester une décision ;
- le coût de maintenance ;
- la facilité de mise à jour ;
- la réversibilité d'une action ;
- la capacité du système à signaler ses limites.

La qualification ajoute les exigences propres au déploiement. Elle examine notamment :

- la finalité du traitement ;
- la provenance des données ;
- leur circulation ;
- leur conservation ;
- les responsabilités ;

- les procédures de recours ;
- le contrôle humain ;
- la gestion des incidents ;
- les effets prévisibles d'une erreur ;
- les obligations sectorielles ;
- les conditions d'arrêt ou de retrait.

La séparation entre relation et décision peut ici devenir une propriété opérationnelle importante. Elle permet d'examiner séparément les informations utilisées par le système et la politique qui transforme ces informations en action. Cette possibilité architecturale doit toutefois être vérifiée dans l'implémentation. Elle ne constitue pas, par sa seule présence théorique, une garantie juridique ou éthique.

é é é é é

Ces évaluations se rencontrent lors de la qualification, mais elles conservent leurs critères propres.

20.25. Du changement d'architecture au changement de paradigme

L'histoire de l'informatique ne peut pas être réduite à une succession de modèles plus performants. Elle peut également être lue comme une succession de niveaux d'organisation :

Cette séquence ne signifie pas que chaque paradigme remplace le précédent. La programmation ne supprime pas le calcul. L'apprentissage ne supprime pas la programmation. L'attention ne rend pas les transformations statistiques inutiles. Un paradigme nouveau réorganise des objets existants et rend certaines

propriétés explicitement manipulables. L'ÉTR ne prétend donc pas abolir les architectures précédentes. Elle propose d'examiner un niveau d'organisation dans lequel le référentiel, la qualification, la relation, la mémoire et le routage ne sont plus seulement des effets internes ou des conventions d'implémentation, mais des objets susceptibles d'être définis, composés, mesurés, modifiés et contestés séparément. La portée de cette proposition dépendra précisément des validations décrites dans ce chapitre. Si cette explicitation n'apporte aucune capacité de mesure, d'intervention ou de contrôle supplémentaire, elle restera une reformulation. Si elle permet au contraire :

- d'isoler des mécanismes auparavant confondus ;
- de conserver des invariants nouveaux ;
- de construire des expériences causales ;
- de localiser les erreurs ;
- de rendre les décisions plus traçables ;

- de réduire le coût de preuve ;
- de séparer les observations des politiques d'action ;

elle pourra être considérée comme un changement de niveau architectural. Le protocole ne doit pas présupposer laquelle de ces conclusions sera obtenue. Il doit rendre possible leur distinction.

Conclusion — Construire une architecture de la preuve

La validation de l'ÉTR ne doit être ni un catalogue de métriques ni une succession de questions générales. Elle doit former une architecture de la preuve. Cette architecture commence par des objets définis, des hypothèses nommées et des affirmations délimitées. Elle précise leur domaine, leur portée, leur degré de généralité, leur précision et les conditions capables de les soutenir ou de les contredire. Elle relie ensuite chaque affirmation à un protocole contractuel, défini par ses entrées, sa version, ses conditions contrôlées, ses opérations, ses témoins, ses mesures, ses seuils, ses conditions d'échec et ses exigences de traçabilité. Elle situe chaque protocole dans un espace expérimental et ne déclare les comparaisons équitables qu'au regard d'un régime d'équivalence explicitement défini. Elle identifie les invariants, classe les variables, vérifie les transformations, contrôle la conformité des implémentations et distingue les observations, les calculs, les interprétations et les causes. Elle transforme ensuite les protocoles en un graphe de dépendances, localise les différentes catégories d'erreurs et mesure non seulement la performance du système, mais également le coût nécessaire pour établir que cette performance mérite confiance. Elle compare les fonctions plutôt que les marques, distingue l'optimisation de la rupture, recherche les causes par ablation, par contrôle négatif et par intervention contrefactuelle, puis soumet ses résultats à la reproduction, à la réplication et à la réfutation. Elle conserve enfin l'ensemble de ces éléments dans un dossier de preuve capable de relier chaque observation à l'affirmation qu'elle soutient, limite, contredit ou laisse indéterminée. Lorsqu'un usage réel est envisagé, elle ajoute les conditions de validation applicative et de qualification juridique, réglementaire, éthique et opérationnelle. La logique normative du chapitre peut être résumée par :

Le protocole ne confère pas directement un label de validité. Il produit un dossier de preuve. Ce dossier modifie le statut d'une affirmation localisée. L'ÉTR ne sera pas scientifiquement défendue en devenant impossible à contredire. Elle le sera en rendant ses propositions suffisamment précises pour qu'un résultat contraire puisse être reconnu, localisé et intégré au développement de l'architecture.

Une théorie ne devient pas durable parce qu'elle affirme davantage que les autres. Elle le devient parce qu'elle distingue avec précision ce qu'elle définit, ce qu'elle démontre, ce qu'elle implémente,

ce qu'elle observe, ce qu'elle suppose encore et les expériences par lesquelles ces différents statuts pourront être établis.

Une architecture computationnelle ne devient pleinement scientifique ni lorsqu'elle produit les meilleurs résultats, ni lorsqu'elle résiste à toute critique, mais lorsqu'elle rend chacune de ses affirmations suffisamment explicite pour que toute communauté indépendante puisse en examiner les fondements, en reproduire les preuves, en discuter les limites et, si nécessaire, en établir la réfutation par séquence.

Une architecture scientifique n'est donc pas seulement un ensemble d'algorithmes. C'est un ensemble d'affirmations suffisamment précises pour que chacune puisse être située, éprouvée, reproduite, discutée, améliorée, limitée ou réfutée indépendamment des autres. Le présent chapitre ne constitue ainsi pas seulement la conclusion méthodologique du traité de l'ÉTR. Il en définit le seuil scientifique : le point à partir duquel l'architecture cesse d'être uniquement décrite pour devenir effectivement

éprouvable.

CHAPITRE XXI — APPLICATIONS COMPLÈTES ET PARCOURS DE BOUT EN BOUT

UNE MÊME ARCHITECTURE POUR PLUSIEURS MONDES

Les chapitres précédents ont isolé les principaux objets de l'ÉTR :

- la lecture ;
- Compose ;
- les référentiels ;
- la qualification ;
- les relations ;
- la mémoire ;
- le routeur ;
- les protocoles de validation.

Il reste maintenant à montrer comment ces objets coopèrent dans des systèmes réels.

L'objectif de ce chapitre n'est plus d'introduire de nouveaux concepts. Il est d'examiner plusieurs applications complètes afin d'observer ce qui demeure stable lorsque changent :

- les données ;
- les modèles ;
- les référentiels ;
- les contraintes ;
- les décisions ;
- les domaines d'usage.

Le noyau général est le suivant :

x

\overset{\Phi}{\longmapsto}

Q_0

\overset{\operatorname{Compose}}{\longmapsto}

Q_s

\overset{\Lambda_R}{\longmapsto}

Q_q

\overset{\rho_R}{\longmapsto}
\mathcal R
\overset{\operatorname{Router}}{\longmapsto}

D

\overset{\operatorname{Update}}{\longmapsto}
\Omega'

}

où :

- x est l'entrée ;
- \Phi est la lecture ;
- Q_0 est l'état initial ;
- Q_s est l'état structuré ;
- \Lambda_R est la qualification dans le référentiel R ;
- Q_q est l'état qualifié ;
- \rho_R produit le profil relationnel ;
- D est la décision ;
- \Omega' est la mémoire mise à jour.

Ce pipeline ne signifie pas que toutes les applications exécutent exactement les mêmes algorithmes.

Il signifie que leurs opérations peuvent être décrites à travers les mêmes fonctions fondamentales.

21.1. Ce qui change et ce qui demeure

D'une application à l'autre, plusieurs éléments changent.

La lecture d'une phrase peut être produite par un tokenizer et un Transformer.

La lecture d'une image peut être produite par un CNN.

La lecture d'un environnement robotique peut résulter d'une fusion de capteurs.

La qualification d'un document juridique n'utilise pas le même référentiel que la qualification d'une publication sociale.

Le routeur d'un moteur de recommandation ne possède pas les mêmes obligations que celui d'un système médical.

Cependant, certains objets demeurent.

Toute application doit déterminer :

. ce qui est reçu ; . comment cela est représenté ; . selon quel référentiel cette représentation est interprétée ; . quelles relations sont calculées ; . quelle politique transforme ces relations en décision ; . ce qui doit être conservé en mémoire ; . comment les résultats pourront être vérifiés.

La continuité de l'ÉTR ne réside donc pas dans l'identité des modèles.

Elle réside dans la stabilité de cette décomposition fonctionnelle.

21.2. Cas I — Interprétation d'une phrase ambiguë

Considérons la phrase :

Le Jaguar accélère à l'approche du virage.

Le terme « Jaguar » peut désigner :

- un animal ;
- une automobile ;
- une équipe ;
- un logiciel ;
- une organisation ;
- un nom propre.

L'ÉTR ne doit pas choisir immédiatement l'une de ces interprétations. Intuitivement, la phrase « un pangolin accélère à l'approche du virage » réduira la complexité.

Elle doit d'abord conserver sa structure.

21.2.1. Entrée

x =

\text{« Le Jaguar accélère à l’approche du virage. »}

La provenance contient notamment :

\Pi_x

= (

\text{source},
\text{date},
\text{langue},
\text{contexte disponible}

)

———

21.2.2. Lecture

La fonction de lecture segmente et représente l’entrée :

Q_0 =

\Phi_{\mathrm{texte}}(x)

Elle peut produire :

- les tokens ;
- les unités lexicales ;
- les positions ;
- les dépendances grammaticales ;
- les marqueurs temporels ou causaux.

Une sortie simplifiée pourrait être :

Q_0 = (

\text{Jaguar},
\text{accélérer},
\text{virage},
\text{relation temporelle}

)

Cette lecture ne désambiguïse pas encore « Jaguar » et le terme pangolin aura une meilleur chance de changer le sens de la phrase

---

21.2.3. Compose

Compose conserve l'ordre et les dépendances :

Q_s =

\operatorname{Compose}

(

\text{Le},
\text{Jaguar},
\text{accélère},
\text{à l'approche},
\text{du virage}

)

Le système obtient une structure dans laquelle :

- Jaguar est le sujet ;
- accélère est l'action ;
- le virage est une situation anticipée ;
- l'action précède ou accompagne l'approche.

La structure pourrait être cohérente dans plusieurs référentiels.

21.2.4. Sélection du référentiel

Le système examine le contexte.

Supposons que la phrase provienne d'un article automobile.

Le sélecteur produit :

R^* =

R_{\mathrm{automobile}}

LES RÉFÉRENTIELS ZOOLOGIQUE ET LOGICIEL RESTENT DISPONIBLES, MAIS INACTIFS

pour la qualification principale.

\ALPHA_{\MATHRM{AUTOMOBILE}}=1

\ALPHA_{\MATHRM{ZOOLOGIQUE}}=0

\ALPHA_{\MATHRM{LOGICIEL}}=0

———

21.2.5. Qualification

La qualification devient :

Q_q =

\Lambda_{R_{\mathrm{automobile}}}(Q_s)

avec :

\text{Jaguar}
\mapsto
\text{véhicule ou marque automobile}

La qualification peut rester partielle si le système ne sait pas encore s'il s'agit d'une marque, d'un modèle ou d'un véhicule particulier.

Elle ne doit pas inventer cette précision.

———

21.2.6. Relations

Le système compare l'état qualifié à sa mémoire :

\mathcal R

=

\rho_{R_{\mathrm{automobile}}}

(Q_q,

\Omega_{\mathrm{automobile}}

)

Le profil peut contenir :

\mathcal R

= (

r_{\mathrm{compatibilité}},

r_{\mathrm{cohérence}},

r_{\mathrm{confiance}},

r_{\mathrm{ambiguïté}}

)

Par exemple :

\mathcal R

= (0{,}96, 0{,}91, 0{,}84, 0{,}18)

Ces valeurs ne sont pas une vérité. Elles représentent des mesures produites selon une méthode déclarée.

———

21.2.7. Routage

La politique peut être :

D =

\begin{cases}

\text{interprétation automobile}

&

r_{\mathrm{compatibilité}}\geq\theta_1

\\

\text{conserver plusieurs hypothèses}

&

r_{\mathrm{ambiguïté}}\geq\theta_2

\\

\text{demander du contexte}

&

\text{sinon}

\END{CASES}

Ici :

D =

\text{retenir l’interprétation automobile}

avec conservation des hypothèses secondaires.

21.2.8. Mise à jour de la mémoire

La mémoire peut enregistrer :

- l’entrée ;
- la structure ;
- le référentiel actif ;
- la qualification retenue ;
- les alternatives rejetées ;
- les relations ;
- la décision.

\Omega'

=

\operatorname{Update}

(

\Omega,

Q_q,

\mathcal R,

D,

\Pi

)

La mémoire ne doit pas enregistrer « Jaguar = automobile » comme vérité universelle.

Elle doit enregistrer :

Dans cette phrase, dans ce contexte et sous ce référentiel, l’interprétation automobile a été retenue.

21.2.9. Ce que montre ce cas

Ce parcours établit la distinction suivante :

\text{structure}

\neq

\text{qualification}

\neq

\text{décision}

La phrase conserve une structure stable.

La qualification dépend du référentiel.

La décision dépend de la politique et du niveau d'ambiguïté.

21.3. Cas II — Recherche documentaire scientifique

Un chercheur demande :

Quels travaux récents étudient la propagation d'erreurs dans les systèmes multi-agents fondés sur des modèles de langage ?

Le problème ne consiste pas uniquement à retrouver des documents contenant les mêmes mots.

Il faut distinguer :

- pertinence thématique ;
- proximité terminologique ;
- récence ;
- fiabilité ;
- méthodologie ;
- rapport direct avec la question.

21.3.1. Entrée et lecture

x =

\TEXT{REQUÊTE DU CHERCHEUR}

La lecture extrait :

Q_0 = (

\text{systèmes multi-agents},

\text{LLM},

\text{propagation d’erreurs},

\text{travaux récents}

)

Le système identifie également la mission :

G =

\text{recherche scientifique}

21.3.2. Compose

Compose construit la structure relationnelle de la requête :

Q_s =

\operatorname{Compose}(Q_0)

Il doit conserver que :

- l’objet étudié est un système multi-agent ;
- les agents utilisent des modèles de langage ;
- la propriété recherchée est la propagation des erreurs ;
- la contrainte temporelle porte sur la récence.

Une recherche par simple collection de mots pourrait retrouver des articles portant sur chacun de ces thèmes séparément.

Compose conserve leur articulation.

21.3.3. Référentiels actifs

Plusieurs référentiels sont nécessaires :

R_{\MATHRM{THÉMATIQUE}}

R_{\MATHRM{TEMPOREL}}

R_{\MATHRM{MÉTHODOLOGIQUE}}

R_{\MATHRM{FIABILITÉ}}

Le référentiel principal peut être :

R_{\MATHRM{SCIENTIFIQUE}}

Les autres deviennent des sous-référentiels ou des dimensions de contrôle.

21.3.4. Qualification des documents

Pour chaque document d_i :

Q_i =

\Phi_{\mathrm{document}}(d_i)

puis :

Q_{q,i} =

\Lambda_{R_{\mathrm{scientifique}}}(Q_i)

La qualification peut extraire :

- question de recherche ;
- méthodes ;
- données ;
- résultats ;
- limitations ;
- année ;
- statut de publication ;
- citations ou provenance.

Un LLM peut contribuer à cette extraction.

Sa sortie reste une qualification candidate, non une preuve.

21.3.5. Profil relationnel

Pour chaque document :

\mathcal R_i

=

\rho_R(Q_s,Q_{q,i})

avec :

\mathcal R_i

= (

r_{\mathrm{thème}},
r_{\mathrm{méthode}},
r_{\mathrm{récence}},
r_{\mathrm{fiabilité}},
r_{\mathrm{couverture}}

)

Deux documents peuvent obtenir des profils très différents.

Document d_1 :

(0{,}95,0{,}42,0{,}88,0{,}51,0{,}76)

Document d_2 :

(0{,}81,0{,}91,0{,}79,0{,}93,0{,}84)

Le premier est plus proche lexicalement.

Le second est plus solide méthodologiquement.

———

21.3.6. Routeur documentaire

La politique scientifique peut imposer :

r_{\mathrm{fiabilité}}
\geq
\theta_f

et :

r_{\mathrm{méthode}}
\geq
\theta_m

avant le classement final.

Le routeur peut produire :

- documents principaux ;
- documents complémentaires ;
- documents exploratoires ;
- documents rejetés ;

- documents à vérifier.

La recherche ne produit donc pas seulement un classement linéaire.

Elle produit une cartographie fonctionnelle des résultats.

21.3.7. Mémoire scientifique

La mémoire conserve :

- la requête ;
- les documents consultés ;
- leurs qualifications ;
- les relations calculées ;
- les contradictions ;
- les versions ;
- les motifs d'exclusion.

Une recherche ultérieure peut réutiliser ce chemin sans confondre les conclusions anciennes avec les documents nouveaux.

21.3.8. Validation

Le système peut être comparé à :

- BM25 ;
- une recherche vectorielle ;
- un système RAG ;
- un moteur hybride.

Les métriques de tâche peuvent inclure :

- précision ;
- rappel ;
- nDCG ;
- taux de documents réellement pertinents.

Les métriques architecturales peuvent inclure :

- traçabilité des résultats ;
- capacité à expliquer le classement ;

- stabilité lors d'un changement de référentiel ;
- coût de preuve ;
- temps de correction d'une qualification erronée.

21.4. Cas III — LLM et production d'une réponse contrôlée

Un utilisateur demande :

Cette clause contractuelle permet-elle au fournisseur de modifier unilatéralement les tarifs ?

Un LLM pourrait répondre directement.

L'ÉTR impose une décomposition plus prudente.

21.4.1. Lecture du texte

La clause contractuelle est lue :

Q_{\mathrm{clause}}

=

\Phi_{\mathrm{LLM}}(x)

La sortie peut contenir :

- parties ;
- modalité de modification ;
- préavis ;
- conditions ;
- droit de résiliation ;
- exceptions.

Le LLM intervient ici comme moteur de lecture et d'extraction.

21.4.2. Compose

Compose conserve l'organisation de la clause :

Q_s =

\operatorname{Compose}

(

\text{condition},

\text{action autorisée},

\text{préavis},

\text{droit du client},

\text{exception}

)

L'ordre juridique importe.

Une exception placée après une autorisation peut en limiter fortement la portée.

21.4.3. Qualification juridique

Q_q =

\LAMBDA_{R_{\MATHRM{JURIDIQUE}}}(Q_S)

La qualification peut distinguer :

- pouvoir de modification ;
- condition de validité ;
- asymétrie contractuelle ;
- obligation d'information ;
- possibilité de refus ;
- incertitude normative.

Le LLM peut proposer cette qualification.

Une règle juridique, une base documentaire ou un professionnel peut la vérifier.

21.4.4. Relations

Le système compare la clause à :

- d'autres clauses ;
- des règles normatives ;
- des décisions antérieures ;
- des modèles contractuels.

\mathcal R

= (

r_{\mathrm{compatibilité}},
r_{\mathrm{risque}},
r_{\mathrm{asymétrie}},
r_{\mathrm{incertitude}}

)

———

21.4.5. Routeur

Le routeur ne doit pas nécessairement produire :

D=\TEXT{OUI}

ou :

D=\TEXT{NON}

Il peut produire :

D=

\text{réponse conditionnelle}

ou :

D=

\text{vérification juridique requise}

Une politique possible :

D =

\begin{cases}
\text{réponse directe}

&

U<\theta_1
\\
\text{réponse conditionnelle}

&

\theta_1\leq U<\theta_2

\\

\text{abstention et recours humain}

&

U\geq\theta_2

\end{cases}

21.4.6. Génération finale

Le LLM peut formuler la réponse à partir du registre décisionnel :

E =

\operatorname{Generate}

(Q_q,

\mathcal R,

D,

\Pi

)

La réponse ne doit pas inventer une justification nouvelle.

Elle doit refléter :

- les éléments effectivement lus ;
- le référentiel appliqué ;
- les incertitudes ;
- les conditions de la décision.

Le LLM devient ici un générateur d'expression, non l'unique autorité de décision.

21.5. Cas IV — Vision industrielle avec CNN

Une caméra inspecte une pièce mécanique.

L'objectif est de détecter une fissure et de décider si la pièce doit être retirée de la chaîne.

21.5.1. Lecture perceptive

Q_0 =

\Phi_{\mathrm{CNN}}(I)

Le CNN produit :

- cartes d'activation ;
- régions candidates ;
- score de fissure ;
- localisation.

Le CNN demeure l'outil spécialisé de lecture visuelle.

21.5.2. Qualification

Le système doit distinguer :

- rayure superficielle ;
- fissure structurelle ;
- défaut d'éclairage ;
- artefact de capteur ;
- motif normal de fabrication.

Q_q =

\Lambda_{R_{\mathrm{inspection}}}(Q_0)

La qualification peut utiliser :

- le CNN ;
- un modèle géométrique ;
- l'historique des pièces ;
- les normes de fabrication ;
- une seconde image.

21.5.3. Relations

Le système compare le défaut aux profils connus :

$$
\mathcal R
= (
r_{\mathrm{forme}},
r_{\mathrm{profondeur}},
r_{\mathrm{localisation}},
r_{\mathrm{criticité}},
r_{\mathrm{confiance}}
)
$$

Une forte similarité visuelle ne suffit pas si la localisation est non critique.

Inversement, un petit défaut situé dans une zone sensible peut exiger l'arrêt.

21.5.4. Routage

$$
D =
\begin{cases}
\text{continuer}
&
r_{\mathrm{criticité}}<\theta_1
\\
\text{inspection secondaire}
&
\theta_1\leq r_{\mathrm{criticité}}<\theta_2
\\
\text{retirer la pièce}
&
r_{\mathrm{criticité}}\geq\theta_2
\end{cases}
$$

Une contrainte dure peut s'ajouter :

$$
c_{\mathrm{sécurité}}=\text{échec}
\Rightarrow
D=\text{retrait}
$$

21.5.5. Action et mémoire

La décision est compilée en commande :

A =

\operatorname{Compile}(D)

Puis la mémoire conserve :

- image ;
- défaut ;
- lot ;
- machine ;
- décision ;
- inspection humaine éventuelle ;
- résultat final.

Cette mémoire peut servir à réévaluer le CNN et les seuils du routeur.

21.5.6. Validation

Le CNN est évalué sur la détection.

La qualification est évaluée sur la distinction des défauts.

Le routeur est évalué sur le coût des erreurs.

Le système complet est évalué sur :

- défauts non détectés ;
- retraits injustifiés ;
- temps d'inspection ;
- traçabilité ;
- capacité de correction.

Une bonne précision du CNN ne suffit donc pas à valider le système industriel complet.

21.6. Cas V — Robotique multimodale

Un robot mobile doit traverser un entrepôt.

Il reçoit :

- images ;
- lidar ;
- ultrasons ;
- odométrie ;
- instructions ;
- état de la carte.

21.6.1. Lectures parallèles

Q_{\MATHRM{VISION}}

=

\Phi_{\mathrm{CNN}}(I)
Q_{\mathrm{lidar}}

=

\Phi_{\mathrm{lidar}}(L)
Q_{\mathrm{audio}}

=

\Phi_{\mathrm{audio}}(A)
Q_{\mathrm{instruction}}

=

\Phi_{\mathrm{langage}}(x)

Chaque lecteur possède son contrat et son incertitude.

21.6.2. Compose multimodal

Les lectures sont composées :

Q_s =

\operatorname{Compose}_{\mathrm{multi}}

(

Q_{\mathrm{vision}},

Q_{\mathrm{lidar}},

Q_{\mathrm{instruction}},

Q_{\mathrm{mémoire}}

)

Compose doit conserver :

- l'origine des signaux ;
- leur temporalité ;
- leur ordre ;
- leurs incertitudes ;
- leurs éventuelles contradictions.

Un obstacle vu par la caméra mais absent du lidar ne doit pas disparaître dans une fusion opaque.

21.6.3. Qualification

Dans le référentiel de navigation :

Q_q =

\Lambda_{R_{\mathrm{navigation}}}(Q_s)

Le système qualifie :

- obstacle fixe ;
- humain ;
- véhicule mobile ;
- zone interdite ;
- bruit de capteur ;
- trajectoire disponible.

21.6.4. Relations

\mathcal R

= (

r_{\mathrm{distance}},

r_{\mathrm{vitesse}},

$$
r_{\mathrm{trajectoire}},
r_{\mathrm{risque}},
r_{\mathrm{confiance}}
$$

)

Le profil peut être calculé pour plusieurs trajectoires candidates.

21.6.5. Routeur de sécurité

Le routeur sélectionne :

- continuer ;
- ralentir ;
- changer de trajectoire ;
- s'arrêter ;
- demander confirmation.

La sécurité peut dominer les autres référentiels :

$$
R_{\mathrm{sécurité}}
\succ
R_{\mathrm{efficacité}}
$$

21.6.6. Mémoire

La mémoire conserve :

- événements ;
- trajectoires ;
- obstacles ;
- erreurs de capteur ;
- décisions ;
- conséquences.

Elle permet ensuite de comparer les situations nouvelles aux précédentes.

21.6.7. Valeur de la séparation

Le CNN ne décide pas seul.

Le capteur ne décide pas seul.

La qualification ne décide pas seule.

La relation de risque ne décide pas seule.

Le routeur applique la politique de sécurité.

Cette séparation permet de modifier la politique sans réentraîner nécessairement les lecteurs perceptifs.

21.7. Cas VI — Cercle et navigation relationnelle

Cercle constitue une application particulièrement importante de l'ÉTR.

L'objectif n'est pas seulement de recommander un contenu proche.

Il est de construire une navigation selon :

- les pintones ;
- la LOI de l'utilisateur ;
- le référentiel actif ;
- la direction ;
- la nouveauté ;
- la mémoire des parcours.

21.7.1. Entrée

Une publication p entre dans le système :

P

Elle contient :

- texte ;
- image ;
- auteur ;
- temporalité ;

- interactions ;
- contexte.

21.7.2. Lecture

Plusieurs lecteurs produisent :

$$Q_{\mathrm{TEXTE}}$$

$$Q_{\mathrm{IMAGE}}$$

$$Q_{\mathrm{INTERACTION}}$$

$$Q_{\mathrm{TEMPS}}$$

Ces lectures sont ensuite composées.

21.7.3. Qualification en pintones

$$Q_q =$$

$$\Lambda_{R_{\mathrm{pintones}}}(Q_s)$$

Le système attribue ou propose des pintones selon :

- contenu ;
- intention ;
- tonalité ;
- structure ;
- relation à d'autres éléments.

Les pintones sont des objets du référentiel, non des étiquettes arbitraires ajoutées après le calcul.

21.7.4. LOI utilisateur

La LOI peut être représentée par :

$$L_u =$$

$$(p_1,p_2,\ldots,p_n)$$

ou par un état relationnel construit à partir de ces pintones.

La LOI exprime une orientation active.

Elle ne constitue pas une identité absolue de l'utilisateur.

21.7.5. Référentiels de navigation

Plusieurs référentiels peuvent être disponibles :

R_{\MATHRM{PROXIMITÉ}}

R_{\MATHRM{EXPLORATION}}

R_{\MATHRM{APPRENTISSAGE}}

R_{\MATHRM{MODÉRATION}}

Le moteur peut désactiver :

R_{\MATHRM{PROXIMITÉ}}

et activer :

R_{\MATHRM{EXPLORATION}}

afin d'introduire une distance contrôlée.

Le changement doit être enregistré.

21.7.6. Profil relationnel

Pour chaque publication candidate :

\mathcal R(p,L_u)

= (

r_{\mathrm{proximité}},
r_{\mathrm{direction}},
r_{\mathrm{nouveauté}},
r_{\mathrm{compatibilité}},
r_{\mathrm{répétition}},

r_{\mathrm{risque}}

)

Le moteur ne recherche pas uniquement la publication la plus proche.

Il peut rechercher :

- la plus proche dans la direction active ;
- la plus nouvelle dans une zone compatible ;
- un pont entre deux clusters ;
- une rupture limitée ;
- un retour vers une ligne mémorielle.

21.7.7. Routeur de navigation

Dans le référentiel de proximité :

D_{\mathrm{prox}}

=

\arg\max_p
r_{\mathrm{proximité}}

sous contraintes.

Dans le référentiel d'exploration :

D_{\mathrm{expl}}

=

\arg\max_p
r_{\mathrm{nouveauté}}

avec :

r_{\mathrm{compatibilité}}
\geq
\theta_c

et :

r_{\mathrm{risque}}
\leq
\theta_r

Le même profil relationnel peut donc produire plusieurs navigations.

21.7.8. Mémoire des parcours

La mémoire conserve :

- publications activées ;
- pintones ;
- transitions ;
- référentiels ;
- directions ;
- exclusions ;
- retours ;
- bifurcations.

\Omega_u

= (

\Gamma_u,

R_u, L_u,

\Pi_u

)

Cette mémoire ne doit pas réduire l'utilisateur à une catégorie fixe.

Elle conserve une trajectoire évolutive.

21.7.9. Validation de Cercle

Les métriques classiques peuvent inclure :

- engagement ;
- temps de consultation ;
- diversité ;
- pertinence perçue.

Mais elles ne suffisent pas.

Il faut aussi mesurer :

- stabilité des pintones ;
- cohérence des transitions ;
- capacité d'exploration ;
- réduction de la répétition ;
- contrôle du changement de référentiel ;
- traçabilité de la navigation ;
- possibilité de contester ou modifier la LOI.

21.8. Cas VII — Cartographie stratégique d'entreprise

Une entreprise souhaite analyser ses risques de dépendance.

Les sources comprennent :

- contrats ;
- fournisseurs ;
- clients ;
- équipes ;
- projets ;
- compétences ;
- flux financiers ;
- documents internes.

21.8.1. Lectures

Chaque source possède une lecture spécialisée :

Q_{\MATHRM{CONTRATS}}

Q_{\MATHRM{FINANCE}}

Q_{\MATHRM{ORGANISATION}}

Q_{\MATHRM{COMPÉTENCES}}

———

21.8.2. Compose organisationnel

Compose construit les dépendances :

Q_s =

\operatorname{Compose}

(

\text{fournisseur},
\text{contrat},
\text{équipe},
\text{projet},
\text{revenu}

)

Il conserve les relations causales et temporelles connues.

———

21.8.3. Référentiels

L'entreprise peut être lue selon :

R_{\MATHRM{FINANCIER}}

R_{\MATHRM{OPÉRATIONNEL}}

R_{\MATHRM{JURIDIQUE}}

R_{\MATHRM{STRATÉGIQUE}}

Une dépendance peut être faible financièrement mais critique opérationnellement.

Les lectures doivent rester séparées.

21.8.4. Relations

Pour un fournisseur :

\mathcal R_f

= (

r_{\mathrm{coût}},
r_{\mathrm{substituabilité}},
r_{\mathrm{criticité}},
r_{\mathrm{concentration}},
r_{\mathrm{risque}}

)

Pour une compétence :

\MATHCAL R_C

= (

r_{\mathrm{rareté}},
r_{\mathrm{centralité}},
r_{\mathrm{transmission}},
r_{\mathrm{dépendance}}

)

21.8.5. Routeur stratégique

Le routeur peut produire :

- surveiller ;
- diversifier ;
- contractualiser ;
- former ;
- documenter ;
- transmettre ;
- accepter, bloquer le risque.

La décision dépend du référentiel principal et de chaque coordonnées de celui-ci car les distances sont présentées tels les résultats d'une distribution multiplicative et absolue.

Une stratégie financière peut privilégier le coût.

Une stratégie de continuité peut privilégier la résilience.

Une stratégie commerciale peut privilégier la radiation.

21.8.6. Mémoire organisationnelle

La mémoire conserve :

- décisions ;
- hypothèses ;
- changements ;
- effets ;
- événements ;
- révisions.

Une décision stratégique peut ensuite être évaluée selon ses conséquences réelles.

L'entreprise ne conserve pas seulement ce qu'elle a décidé.

Elle conserve pourquoi elle l'a décidé et ce qui s'est produit ensuite.

21.9. Cas VIII — Apprentissage adaptatif et Future of Work

Un système d'apprentissage doit proposer un parcours à une personne.

Il dispose de :

- compétences déclarées ;
- productions ;
- évaluations ;
- intérêts ;
- rythme ;
- objectifs ;
- historique.

21.9.1. Lecture du profil

Q_0 =

\Phi_{\mathrm{apprentissage}}(x)

La lecture ne doit pas réduire la personne à une note globale.

Elle peut produire :

Q_0 = (

\text{compétences},
\text{préférences},
\text{progression},
\text{obstacles},
\text{objectifs}

)

21.9.2. Référentiels

Plusieurs référentiels peuvent être actifs :

R_{\MATHRM{COMPÉTENCES}}

R_{\MATHRM{PÉDAGOGIQUE}}

R_{\MATHRM{MOTIVATION}}

R_{\MATHRM{EMPLOI}}

Le même exercice peut être agencé selon la progression pédagogique et non attribuable selon l'objectif professionnel immédiat.

21.9.3. Qualification

Le système qualifie :

- acquis ;

- lacunes ;
- prérequis ;
- style d’apprentissage ;
- difficulté adaptée ;
- niveau d’incertitude.

Cette qualification doit rester révisable.

Une erreur ponctuelle ne doit pas devenir une identité durable.

21.9.4. Relations

Chaque ressource pédagogique reçoit :

$$\mathcal R_i = (r_{\mathrm{prérequis}}, r_{\mathrm{difficulté}}, r_{\mathrm{objectif}}, r_{\mathrm{nouveauté}}, r_{\mathrm{charge}})$$

21.9.5. Routeur pédagogique

Le routeur peut :

- proposer ;
- différer ;
- réviser ;
- répéter ;
- diversifier ;
- demander une évaluation ;
- solliciter un mentor.

L’option la plus proche n’est pas toujours la meilleure.

Le système peut choisir une ressource légèrement distante afin de construire des liens.

21.9.6. Mémoire et progression

La mémoire conserve :

- parcours ;
- essais ;
- erreurs ;
- corrections ;
- compétences ;
- contexte ;
- temps ;
- stratégies efficaces.

Elle doit distinguer :

$$\TEXT{PERFORMANCE PONCTUELLE}$$

de :

$$\TEXT{COMPÉTENCE CONSOLIDÉE}$$

21.9.7. Risque

Un système adaptatif peut enfermer une personne dans un profil.

L'ÉTR doit donc permettre :

- plusieurs référentiels ;
- la révision des qualifications ;
- l'exploration ;
- la contestation ;
- les régénérations et gestion de certaines mémoires ;
- la conservation de l'incertitude.

21.10. Cas IX — Système multi-agent scientifique

Une équipe d'agents doit produire une revue de littérature.

Agents :

A_{\MATHRM{RECHERCHE}}

A_{\MATHRM{EXTRACTION}}

A_{\MATHRM{MÉTHODE}}

A_{\MATHRM{CONTRADICTION}}

A_{\MATHRM{SYNTHÈSE}}

21.10.1. Routage des tâches

Le routeur attribue :

D_i =

\operatorname{RouteTo}(a_i)

selon :

- compétence ;
- coût ;
- disponibilité ;
- Hiérarchique/permission/responsabilité;
- fiabilité.

21.10.2. Sorties comme qualifications candidates

Chaque agent produit :

Q_I

avec :

- provenance ;

- modèle ;
- prompt ;
- sources ;
- incertitude.

Aucune sortie ne devient automatiquement une mémoire validée.

21.10.3. Relations entre sorties

Le système compare :

$$\RHO(Q_I,Q_J)$$

afin de détecter :

- accord ;
- complémentarité ;
- conflit ;
- redondance ;
- dépendance à une même source.

Trois agents répétant le même article ne constituent pas trois validations indépendantes.

21.10.4. Agent de contradiction

L'agent de contradiction cherche :

- hypothèses opposées ;
- résultats incompatibles ;
- faiblesses méthodologiques ;
- sources manquantes.

Il ne doit pas seulement critiquer la synthèse finale.

Il intervient avant la décision.

21.10.5. Synthèse

Le routeur décide :

- quelles propositions sont retenues ;
- lesquelles restent ouvertes ;
- lesquelles exigent une vérification ;
- lesquelles doivent être exclues.

La synthèse finale conserve :

- consensus ;
- dissensus ;
- provenance ;
- incertitude ;
- statut des affirmations.

21.10.6. Mémoire collective

La mémoire commune doit enregistrer :

- contributions ;
- transformations ;
- conflits ;
- validations ;
- décisions.

Elle ne doit pas effacer la provenance individuelle.

21.11. Cas X — Application complète intégrale

Considérons maintenant un cas unique suivi de bout en bout.

Une publication apparaît sur Cercle :

Une nouvelle méthode permettrait de réduire de moitié la consommation énergétique des centres de données.

L'objectif du système est de déterminer :

- comment qualifier cette publication ;
- où la situer ;
- à qui la présenter ;
- avec quel niveau de prudence ;
- ce qui doit être inscrit en mémoire.

21.11.1. Entrée

x =

\text{publication brute}

La provenance contient :

\Pi_x

= (

\text{auteur},
\text{date},
\text{support},
\text{liens},
\text{médias}

)

21.11.2. Lecture multimodale

Q_{\mathrm{texte}}

=

\Phi_{\mathrm{texte}}(x)
Q_{\mathrm{source}}

=

\Phi_{\mathrm{documentaire}}(x)
Q_{\mathrm{image}}

=

\Phi_{\mathrm{vision}}(x)

La lecture textuelle extrait :

- méthode nouvelle ;
- réduction annoncée ;
- centres de données ;
- consommation énergétique.

La lecture documentaire identifie :

- présence ou absence de source ;
- nature de la source ;
- date ;
- auteur ;
- publication scientifique éventuelle.

21.11.3. Compose

Q_s =

\operatorname{Compose}

(

Q_{\mathrm{texte}},
Q_{\mathrm{source}},
Q_{\mathrm{image}}

)

Compose conserve que :

- la réduction est une affirmation ;
- la valeur annoncée est « moitié » ;
- l'objet est la consommation énergétique ;
- le domaine est celui des centres de données ;
- la preuve n'est pas encore établie.

21.11.4. Référentiels

Plusieurs référentiels peuvent être mobilisés :

R_{\MATHRM{ÉNERGIE}}

R_{\MATHRM{SCIENCE}}

R_{\MATHRM{INNOVATION}}

R_{\MATHRM{FIABILITÉ}}

R_{\MATHRM{NAVIGATION}}

Le référentiel scientifique qualifie l'état de preuve.

Le référentiel d'innovation qualifie la nouveauté.

Le référentiel de navigation détermine les utilisateurs susceptibles d'être intéressés.

———

21.11.5. Qualification

Le système peut produire :

Q_q =

(

\text{affirmation technique},
\text{forte amplitude},
\text{source insuffisamment établie},
\text{innovation potentielle},
\text{vérification requise}

)

Le terme « permet » ne doit pas être transformé en résultat établi.

La modalité « permettrait » indique déjà une réserve.

———

21.11.6. Relation aux pintones

La publication peut être positionnée relativement à des pintones comme :

- innovation énergétique ;
- infrastructure numérique ;
- sobriété ;
- centres de données ;

- recherche appliquée ;
- transition industrielle.

La relation peut être :

\mathcal R_{\mathrm{pintones}}

= (

r_1,\ldots,r_n

)

———

21.11.7. Relation à la LOI utilisateur

Pour un utilisateur orienté vers :

L_u = (

\text{IA},
\text{énergie},
\text{infrastructure}

)

le profil peut être :

\mathcal R_u

= (

r_{\mathrm{proximité}}=0{,}89,
r_{\mathrm{nouveauté}}=0{,}76,
r_{\mathrm{fiabilité}}=0{,}41,
r_{\mathrm{compatibilité}}=0{,}93

)

———

21.11.8. Routage

Dans un référentiel de proximité, la publication pourrait être sélectionnée.

Mais le faible niveau de fiabilité peut imposer une qualification visible :

D =

\text{afficher avec statut « affirmation à vérifier »}

Une politique plus stricte pourrait décider :

D =

\text{ne pas recommander comme connaissance établie}

Les deux décisions utilisent le même profil relationnel.

21.11.9. Recherche documentaire secondaire

Le routeur peut déclencher :

D_2 =

\operatorname{VERIFY}

Le système recherche :

- article source ;
- prépublication ;
- brevet ;
- données ;
- réplication ;
- analyse indépendante.

De nouvelles preuves peuvent modifier :

$$\operatorname{STAT}(\mathcal{C})$$

de l'affirmation :

La méthode réduit de moitié la consommation.

21.11.10. Mise à jour

La mémoire conserve séparément :

. la publication ; . l'affirmation ; . la qualification initiale ; . les relations ; . la décision de navigation ; . les sources retrouvées ; . le statut actualisé.

La publication ne doit pas être réécrite après la vérification.

Son affirmation doit être reliée au nouveau dossier de preuve.

21.11.11. Évolution

Supposons qu'une étude indépendante conclue à une réduction moyenne

de 18\%.

La mémoire conserve :

\mathcal C_1

=

\text{réduction annoncée de }50\%
\mathcal C_2

=

\text{réduction observée de }18\%

avec :

\mathcal C_2
\overset{\text{limite}}{\longrightarrow}
\mathcal C_1
Le système ne remplace pas silencieusement 50\% par 18\%.

Il conserve l'histoire de l'affirmation.

21.11.12. Validation complète

Ce cas peut être utilisé pour évaluer :

- la lecture de la modalité ;
- Compose ;
- la sélection des référentiels ;
- la qualification ;
- les relations ;
- le routeur ;
- la mémoire ;
- le dossier de preuve ;
- la mise à jour.

Le pipeline complet devient :

\begin{aligned}

x

&\overset{\Phi}{\longmapsto}

Q_0

\\

&\overset{\operatorname{Compose}}{\longmapsto}

Q_s

\\

&\overset{\Lambda_R}{\longmapsto}

Q_q

\\

&\overset{\rho_R}{\longmapsto}
\mathcal R

\\

&\overset{\operatorname{Router}}{\longmapsto}

D

\\

&\overset{\operatorname{Update}}{\longmapsto}
\Omega'

\\

&\overset{\mathcal P}{\longmapsto}
\mathcal D_{\mathrm{preuve}}

\\

&\longrightarrow
\operatorname{Stat}(\mathcal C).

\end{aligned}

}

Le protocole de validation n’est donc pas extérieur au parcours.

Il accompagne la chaîne et modifie le statut des affirmations produites ou rencontrées.

21.12. Le rôle des architectures existantes

Les applications précédentes montrent que l'ÉTR n'exige pas la disparition des modèles existants.

Elle peut utiliser :

- un Transformer pour lire ou qualifier un texte ;
- un CNN pour lire une image ;
- un GNN pour propager des informations sur un graphe ;
- BM25 pour produire des candidats documentaires ;
- une base vectorielle pour rechercher des proximités ;
- un moteur symbolique pour vérifier des règles ;
- un humain pour valider une décision.

On peut écrire :

\Phi

=

\Phi_{\mathrm{CNN}}
\Lambda

=

\Lambda_{\mathrm{LLM}}
\rho

=

\rho_{\mathrm{embedding}}

D =

\operatorname{Router}_{\mathrm{règles}}

L'ÉTR ne définit pas toujours l'algorithme interne de chaque fonction.

Elle définit leur place, leur contrat, leur relation et leurs effets.

21.13. Ce que l'ÉTR unifie réellement

L'ÉTR n'unifie pas toutes les données en une représentation unique.

Elle n'unifie pas toutes les relations en une métrique unique.

Elle n’unifie pas toutes les décisions en une politique unique.

Elle unifie la manière de décrire leur articulation.

Cette distinction est essentielle.

L’unification porte sur :

\TEXT{LES RÔLES}

\TEXT{LES CONTRATS}

\TEXT{LES TRANSFORMATIONS}

\TEXT{LES PROVENANCES}

\TEXT{LES STATUTS}

Elle ne porte pas sur l’effacement des différences entre domaines.

21.14. Limites applicatives

L’usage de l’ÉTR peut introduire plusieurs coûts.

La séparation des objets augmente :

- le nombre de composants ;
- la quantité de métadonnées ;
- le coût de traçage ;
- la complexité de gouvernance ;
- le temps de conception des référentiels.

Une architecture ÉTR mal définie peut devenir :

- bureaucratique ;
- excessivement lourde ;
- faussement explicite ;
- dépendante de référentiels arbitraires ;
- difficile à maintenir.

La présence de traces ne garantit pas leur qualité.

La déclaration d’un référentiel ne garantit pas sa validité.

La séparation des fonctions ne garantit pas qu’elles soient correctement implémentées.

L'ÉTR apporte donc une discipline.

Elle ne remplace ni le travail scientifique ni le travail d'ingénierie.

21.15. Conditions de déploiement

Avant tout déploiement, le système doit répondre à plusieurs questions.

QUELS OBJETS SONT RÉELLEMENT NÉCESSAIRES ?

QUELS RÉFÉRENTIELS DOIVENT ÊTRE ACTIFS ?

QUELLES TRANSFORMATIONS DOIVENT ÊTRE INSTRUMENTÉES ?

QUELLES DÉCISIONS EXIGENT UN HUMAIN ?

QUELLES INFORMATIONS DOIVENT ÊTRE MÉMORISÉES ?

QUELLES INFORMATIONS NE DOIVENT PAS L'ÊTRE ?

QUELS PROTOCOLES DOIVENT ÊTRE EXÉCUTÉS EN CONTINU ?

QUELS ÉVÉNEMENTS IMPOSENT UNE ABSTENTION OU UN ARRÊT ?

Une architecture complète n'est pas nécessairement une architecture maximale.

La bonne implémentation est celle qui conserve les distinctions utiles au problème étudié sans ajouter des objets sans fonction.

21.16. Une architecture comme système évolutif

L'ÉTR permet d'envisager des systèmes dans lesquels :

- les lecteurs peuvent être remplacés ;
- les référentiels peuvent évoluer ;
- les qualifications peuvent être révisées ;
- les politiques peuvent changer ;
- la mémoire peut être corrigée ;
- les protocoles peuvent être renforcés.

Cette évolution ne doit pas obligatoirement détruire l'ensemble du système.

Chaque changement peut être localisé.

$$
\Delta
$$

$$
= (
$$

$$
\text{objet modifié},
$$

$$
\text{dépendances},
$$

$$
\text{protocoles affectés},
$$

$$
\text{preuves à réviser}
$$

$$
)
$$

L'architecture devient ainsi révisable sans devenir indéterminée.

21.17. Le noyau commun

À travers le langage, la vision, la robotique, la recherche, l'entreprise, l'apprentissage et les réseaux sociaux, la même structure réapparaît :

$$
\text{Lire}
$$

$$
\rightarrow
$$

$$
\text{construire}
$$

$$
\rightarrow
$$

$$
\text{qualifier}
$$

$$
\rightarrow
$$

$$
\text{relier}
$$

$$
\rightarrow
$$

$$
\text{décider}
$$

$$
\rightarrow
$$

$$
\text{mémoriser}
$$

$$
\rightarrow
$$

$$
\text{éprouver}
$$

$$
\}
$$

Chaque domaine remplit ces fonctions avec ses propres outils.

Un CNN ne devient pas un LLM.

Un référentiel juridique ne devient pas un référentiel pédagogique.

Une décision robotique ne devient pas une recommandation sociale.

Mais leurs transformations peuvent être décrites selon une même architecture fonctionnelle.

Conclusion générale — Une architecture, plusieurs mondes

L'ÉTR est née d'une question :

Comment représenter et exploiter des transformations relationnelles sans confondre les objets, les interprétations, les relations, les décisions et la mémoire ?

La réponse proposée par cet ouvrage ne consiste pas en une équation unique destinée à remplacer toutes les autres.

Elle consiste en une architecture.

Cette architecture distingue :

\TEXT{L'OBJET}

de :

\TEXT{SA LECTURE}

la lecture de :

\TEXT{SA QUALIFICATION}

la qualification de :

\TEXT{SA RELATION AUX AUTRES ÉTATS}

la relation de :

\TEXT{LA DÉCISION QUI L'EXPLOITE}

et la décision de :

\TEXT{LA MÉMOIRE QUI EN CONSERVE LES EFFETS}

Cette séparation ne fragmente pas artificiellement le système.

Elle rend ses transformations observables.

Elle permet de modifier un référentiel sans réécrire la structure.

Elle permet de changer une politique sans falsifier les relations.

Elle permet de corriger une qualification sans effacer l'entrée.

Elle permet de réviser une mémoire sans perdre l'histoire.

Elle permet enfin de construire des protocoles capables d'atteindre des affirmations localisées plutôt que de prononcer des verdicts globaux.

L'ÉTR ne prétend pas que tout phénomène puisse être entièrement rendu explicite.

Elle affirme qu'un système devient plus gouvernable lorsque les transformations qui peuvent être déclarées le sont effectivement.

Les architectures existantes conservent donc leur rôle.

Les Transformers restent des instruments puissants de représentation et de génération contextuelles.

Les CNN restent adaptés à l'apprentissage de structures locales.

Les GNN restent adaptés à la propagation sur graphes.

Les moteurs documentaires restent efficaces pour la recherche.

Les agents restent capables de distribuer les tâches et les actions.

L'ÉTR ne les remplace pas.

Elle propose un langage permettant de les situer dans une chaîne commune, de déclarer leurs responsabilités et de conserver les conditions de leurs productions.

Le résultat peut être résumé par le pipeline général :

x

\overset{\Phi}{\longmapsto}

Q_0

\overset{\operatorname{Compose}}{\longmapsto}

Q_s

\overset{\Lambda_R}{\longmapsto}

Q_q

\overset{\rho_R}{\longmapsto}
\mathcal R
\overset{\operatorname{Router}}{\longmapsto}

D

\overset{\operatorname{Update}}{\longmapsto}
\Omega'

}

auquel s'ajoute la chaîne scientifique :

\mathcal C

\overset{\mathcal P}{\longrightarrow}

\mathcal D_{\mathrm{preuve}}

\longrightarrow

\operatorname{Stat}(\mathcal C)

}

La première décrit comment un système transforme une entrée.

La seconde décrit comment les affirmations portant sur ce système deviennent éprouvables.

Ensemble, elles définissent la proposition essentielle du traité :

\text{une architecture relationnelle ne doit pas seulement produire des

résultats ;} }

\text{elle doit permettre de situer, comprendre, contrôler et réviser les

transformations qui les produisent.} }

L'unité de l'ÉTR ne réside donc pas dans l'uniformité des mondes auxquels elle s'applique.

Elle réside dans la stabilité des questions qu'elle impose à chacun d'eux :

- Qu'est-ce qui a été lu ?
- Qu'est-ce qui a été construit ?
- Dans quel référentiel ?
- Quelle qualification a été produite ?
- Quelle relation a été calculée ?
- Quelle politique a décidé ?
- Quelle mémoire a été modifiée ?
- Quelle preuve permet de soutenir cette transformation ?

Si ces questions peuvent recevoir une réponse précise, l'architecture technologique et mathematique devient proche de chaque chaque élements envisagés et synergiques à l'utilisateur, en ne devenant plus une succession opaque de calculs.

Elle devient un système relationnel stable, combinatoire et dynamique exprimé dans ce livre : très proche du résultats d'une architecture cognitive biologique : examinable, mesurable, composable et révisable.

C'est au titre et au seuil suivant que s'achève cet ouvrage.

Ses objets, ses relations, ses applications et les conditions de leur mise en œuvre et à l'épreuve disposent désormais d'un langage commun à partir duquel la recherche peut réellement commencer.