Validation temporelle explicable de faits par la découverte de contraintes temporelles complexes dans les graphes de connaissances

Thibaut Soulard, Joe Raad, Fatiha Saïs

Validation temporelle explicable de faits par la découverte de contraintes temporelles complexes dans les graphes de connaissances

Thibaut Soulard, Joe Raad, Fatiha Saïs
Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique
91190 Gif-sur-Yvette, France

prénom.nom@lisn.fr

Résumé

La question de la détection des fausses nouvelles a pris de l'importance avec la diffusion croissante de flux d'informations non vérifiées sur le Web. Pour relever ce défi, les graphes de connaissances (GC) sont un moyen efficace pour vérifier le contenu des informations diffusées grâce aux faits structurés qu'ils contiennent. Toutefois, la validité de certains faits est dépendante d'un certain contexte temporel. Cette validation temporelle est une question qui n'a pas encore reçu beaucoup d'attention dans la littérature. Notre travail introduit une nouvelle approche interprétable et explicable qui exploite la puissance des graphes de connaissances pour classer les faits en évaluant leur validité ou leur réfutation dans un intervalle temporel. Nous avons dévelopé une symbolique qui est fondée sur les relations d'Allen entre les intervalles temporels et qui étend ces relations aux séquences temporelles. Nous avons évalué notre approche sur l'un des plus grands graphes de connaissances disponibles publiquement et comparons sa performance à travers de multiples hyper-paramètres avec une variante neuro-symbolique que nous avons aussi proposé.

Mots-clés

Graphe de connaissances, Véracité, Séquence temporelle, IA Hybride

Abstract

The issue of fake news has gained significance in light of the escalating flow of unverified information among users. To face this challenge, Knowledge Graphs (KGs) are a means for verifying news content from statements in the KG. However, an aspect that has yet to receive attention is the temporal validation of these statements (i.e. verifying whether a statement is true in a given time interval). Our work introduces a new interpretable and explainable approach that leverages the power of KGs to classify user-inputted facts by assessing their temporal validity or refutation. We developed a symbolic approach that is based on Allen's relations between temporal intervals and extends these relations to time sequences. We test our symbolic framework on one of the largest publicly available KGs and compare its performance across multiple hyper-parameters with the neuro-symbolic extension that we also developed.

Keywords

Knowledge Graph, Veracity, Temporal Aequence, Hybrid AI

1 Introduction

Les graphes de connaissances sont devenus des sources de données cruciales car ils soutiennent le développement de nombreuses applications liées à l'intelligence artificielle. En s'appuyant sur un langage simple et standardisé tel que RDF[1], les graphes de connaissances peuvent représenter des faits et des connaissances complexes sur le monde sous la forme de triplets : `<sujet, propriété, objet>`. Cependant, malgré le grand nombre de graphes de connaissances publiés ces dernières années, seuls quelques-uns associent explicitement une composante temporelle à leurs faits dépendant du temps. Cette composante temporelle, incluse par exemple dans Wikidata[2] -l'une des plus grandes bases de données publiques, est essentielle pour la qualité des bases de données, car de nombreux faits ne sont valables que dans un intervalle de temps spécifique ou à un moment précis. Par exemple, `<Obama, headOfState, USA>` n'est valable que dans l'intervalle de temps commençant à `2009/01/20` et se terminant à `2017/01/20`.
Ces graphes de connaissances étant de plus en plus utilisés pour des applications de vérification des faits, il devient nécessaire de s'assurer de la validité temporelle de leurs faits. Dans ce travail, nous proposons une approche explicable pour vérifier si un fait est valide dans un intervalle de temps donné. Cette approche peut être appliquée à tout graphe de connaissance associant un intervalle temporel à ses faits sans aucune connaissance préalable des données. Nous présentons d'abord les travaux connexes (section 2) et introduisons les notations nécessaires (section 3). Nous présentons ensuite notre approche pour découvrir les contraintes d'ordre temporel dans les données (section 4), telles que la contrainte selon laquelle un député doit être élu *Before*

1. Resource Description Framework : https://www.w3.org/RDF/
2. https://www.wikidata.org/

d'occuper sa fonction. Ensuite, nous étendons et combinons les contraintes découvertes pour prendre une décision sur la validité temporelle d'un fait donné (section 5). Nous explorons différentes méthodes de combinaison, telles que le traitement égal de toutes les contraintes en utilisant un système de vote pour évaluer la validité d'un fait, ou l'entraînement d'un modèle d'apprentissage automatique pour apprendre quelles contraintes temporelles sont les plus appropriées pour valider un fait. Enfin, nous testons notre approche sur différentes classes, contenant des millions de faits temporels, dans le graphe de connaissance Wikidata (section 6).

2 État de l'art

Bien que les questions liées à la spécification du temps de validité ou de la portée temporelle dans la communauté des bases de données relationnelles soient bien connues [9], ce n'est que ces dernières années que ce sujet a pris une importance particulière dans le développement des graphes de connaissances [3, 13]. Lorsque ces graphes prennent en compte la dynamicité des prédicats de faits dont la vérité est une fonction du temps, un fait est alors considéré comme vrai dans un laps de temps donné. Ce type d'informations sur la dynamicité des prédicats, lorsqu'elles sont disponibles, peut améliorer les résultats des approches de complétion des graphes de connaissances [11] qui peuvent tirer parti de la portée temporelle pour désambiguïser les différents candidats possibles. Cependant, ces approches dépendent à la fois de la disponibilité et de la validité de ces informations temporelles dans le graphe de connaissances et de leur validité. C'est pourquoi certaines approches ont été conçues pour générer de nouvelles informations temporelles, comme décrit dans [11]. Ces approches peuvent s'intéresser soit à la tâche d'interpolation, où l'on peut compléter des faits qui se sont déroulés dans le passé, soit à une tâche d'extrapolation où l'on peut chercher à prédire l'apparition de nouveaux faits.

Comme présenté dans l'étude [11], les approches de complétion GC existantes qui s'intéressent aux informations temporelles utilisent diverses techniques telles que celles basées sur l'apprentissage profond ou les approches neuro-symboliques. Les approches basées sur l'apprentissage profond peuvent utiliser *la traduction dans un espace vectoriel*, *la décomposition du tenseur*, *GCN*, *LSTM*, *GRU* pour apprendre un modèle de prédiction. Les approches neuro-symboliques telles que [14] qui sont à la fois basées sur la connaissance du domaine et l'apprentissage automatique utilisent des contraintes temporelles pour injecter la connaissance du domaine dans le modèle de prédiction. Cette approche améliore TTransE [7] et TA-TransE [6] grâce à l'introduction d'un *ordre temporel* (par exemple, `wasBornIn` $\leq$ `diedIn`) pour les prédicats de la même entité et *disjointure temporelle* (par exemple, dans les pays monogames, une personne ne peut pas être mariée à deux individus différents au cours du même intervalle temporel). Dans [4], les auteurs exploitent l'interaction entre les intervalles temporels des prédicats pour la même entité mais s'appuient sur *Markov Logic Networks* et *Probalistic Soft Logics* pour résoudre la tâche finale.

En ce qui concerne la découverte de contraintes temporelles, il existe dans les bases de données relationnelles, et plus particulièrement dans le domaine du profilage des données [1], certaines approches qui découvrent des relations d'ordre entre les attributs, telles que [5, 12]. Mais les contraintes découvertes n'impliquent que des opérateurs arithmétiques (c'est-à-dire $\leq, <$) et n'utilisent pas d'opérateurs dédiés aux intervalles. A notre connaissance, il n'existe aucune approche permettant de découvrir des contraintes temporelles avec l'expressivité présentée dans ce travail, et il n'existe aucune approche traitant du problème de la validation de l'information temporelle en combinant et en exploitant des contraintes temporelles aussi expressives.

3 Préliminaires

Notre approche de la validation des faits temporels est conçue pour être appliquée aux graphe de connaissances temporels et repose sur l'algèbre d'Allen [2] pour la comparaison des intervalles de temps. Dans cette section, nous présentons le contexte préliminaire et introduisons les concepts et les notations nécessaires.

3.1 Algèbre d'Allen

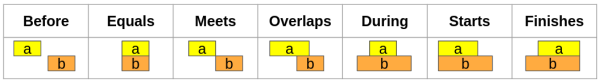

FIGURE 1 – Les 7 relations atomiques d'Allen

L'algèbre d'intervalles d'Allen est une référence pour la représentation des relations entre les intervalles de temps. Elle est composée de treize relations de base qui sont distinctes (c'est-à-dire qu'au plus une relation peut être énoncée pour une paire d'intervalles), exhaustives (c'est-à-dire que chaque paire d'intervalles peut être décrite par l'une des treize relations) et qualitatives (c'est-à-dire qu'aucune durée numérique n'est prise en compte). Ces treize relations peuvent être réduites aux sept relations atomiques présentées dans la figure 1.

Definition 1 (L'ensemble des relations d'Allen)
L'ensemble des relations d'Allen peut être divisé en deux groupes différents : relations disjoints (DA) représentant l'ensemble des relations où les intervalles sont disjoints, et relations intersectés (IA) représentant les relations où les intervalles impliquent une intersection temporelle :
– $DA = \{Before, Meets\}$,
– $IA = \{Equals, Overlaps, During, Starts, Finishes\}$.

Pour comparer deux intervalles de temps $I1$ et $I2$, nous pouvons soit spécifier la relation atomique entre eux, soit la généraliser en spécifiant s'ils sont disjoints ou s'ils ont une intersection. Par exemple, *before(I1, I2)* peut être généralisé en *DA(I1, I2)*.

3.2 Graphe de connaissances temporels

Dans ce travail, nous nous concentrons sur les GC temporels, c'est-à-dire les GC qui associent certains de leurs faits à une information temporelle pour exprimer l'intervalle de temps pendant lequel un fait est valide. Nous définissons tout d'abord les graphe de connaissances RDF, puis les graphe de connaissances temporels.

Definition 2 (Grpahe de connaissance RDF) *Nous considérons un graphe de connaissances défini par une paire $(\mathcal{O}, \mathcal{G})$, où :*

- *$\mathcal{O} = (\mathcal{C}, \mathcal{P})$ est une ontologie représentée en OWL et composée d'un ensemble de classes $\mathcal{C}$ et de propriétés $\mathcal{P}$.*
- *$\mathcal{G}$: est un graphe de données RDF, composé d'un ensemble de faits représentés par des triplets de la forme {(s, p, o) | s $\in \mathcal{I}$, p $\in \mathcal{P}$, o $\in \mathcal{I} \cup \mathcal{L}$}, où $\mathcal{I}$ est l'ensemble des entités (IRIs), $\mathcal{P}$est l'ensemble des propriétés, et $\mathcal{L}$ est l'ensemble de litéraux (tel que des nombres ou des chaînes de caractères).*

Dans un graphe de connaissances composé d'un ensemble de triplets <s, p, o>, on peut distinguer trois types de faits :

- **Faits temporels concrets :** dont l'objet est de type `xsd:date` et dont la validité est illimitée dans le temps, comme : `<Mozart, birthDate, "1756/01/27">`.
- **Faits tautologiques :** vrai pendant toute la durée de vie d'une entité et dont le prédicat n'est pas sensible au temps, comme : `<Mozart, birthPlace, Salzbourg>`.
- **Faits dépendant du temps :** dont la validité est limitée à un intervalle de temps et dont le prédicat est sensible au temps, comme : `<Obama, headOfState, USA>`.

Dans ce travail, nous nous concentrons sur *faits dépendant du temps* qui sont associés à une composante temporelle, en plus de *faits temporels concrets* qui aident à générer des contraintes temporelles.

Definition 3 (Graphe de connaissances temporels) *Nous définissons un graphe de connaissances temporel $\mathcal{TKG}$ comme un ensemble de quadruplets sous la forme de (s, p, o, t), qui étend les triples du graphe de données RDF en ajoutant la composante temporelle t exprimant la validité temporelle du fait qui peut être un horodatage ou un intervalle de temps.*

Nous considérons que tout marqueur temporel t peut être représenté par un intervalle de temps $[t; t + \epsilon]$, ϵ étant une durée insignifiante qui peut être déterminée en fonction de la granularité temporelle considérée (par exemple, des siècles, des années, des jours, des minutes). Par conséquent, dans le reste du document, nous nous référons à l'information temporelle en tant qu'intervalle de temps. Nous désignons par $\mathcal{T}$ l'ensemble de tous les intervalles utilisés dans TKG, chaque intervalle $I \in \mathcal{T}$ ayant une date de début et une date de fin, notées respectivement $I.s$ et $I.e$.

4 Découverte de contraintes temporelles

Pour valider un fait dans un graphe de connaissances temporelles TKG, notre approche consiste à vérifier si l'information temporelle encodée pour ce fait est cohérente par rapport à une liste de contraintes temporelles. Ces contraintes peuvent exprimer soit une disjonction, soit une intersection entre les intervalles temporels des faits (voir la définition 1). Par exemple, une contrainte temporelle exprimant la disjonction peut indiquer qu'un président américain doit être élu *Before* d'occuper sa fonction, sous la forme `Before(elected, headOfState)`. Cependant, de telles contraintes temporelles sont rarement incluses dans le TKG et sont difficiles à collecter manuellement. C'est pourquoi, dans une première étape de notre approche, nous introduisons une nouvelle méthode pour découvrir ce type de contraintes temporelles à partir du TKG. Notre approche consiste d'abord à découvrir toutes les contraintes temporelles pour une seule entité, soit des contraintes simples qui peuvent être exprimées en utilisant les relations d'Allen (section 4.2), soit des contraintes complexes (section 4.3). Ensuite, les contraintes découvertes sont évaluées et généralisées à toutes les entités du TKG du même type (section 4.4).

4.1 Définition et comparaison des séquences temporelles

Pour chaque entité du TKG, nous pouvons construire une séquence temporelle pour chaque propriété dépendant du temps et décrivant l'entité dans le graphe (voir la définition 4). Par abus de langage, nous utiliserons parfois la formulation *un quadruplet se produit dans un intervalle de temps* pour faire référence au fait que le fait est valide dans l'intervalle de temps.

Definition 4 (Séquence temporelle) *La séquence temporelle d'une entité x pour une propriété p est l'ensemble ordonné des intervalles S des quadruplets $\{q_1, \ldots, q_n\}$, sous la forme de $< x, p, y_k, I_k >$ avec I_1 ayant la date de début la plus ancienne et I_n ayant la date de début la plus tardive dans la séquence temporelle.*

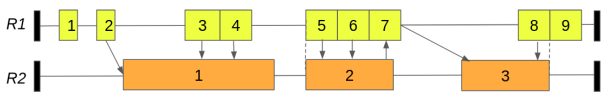

FIGURE 2 – Exemple d'une paire de séquences temporelles contenant 9 et 3 intervalles de temps. Les flèches renvoient à des comparaisons entre séquences.

La figure 2 présente les séquences temporelles de deux propriétés $R1$ et $R2$ pour une seule entité, chaque élément de la séquence représentant un intervalle de temps. Afin de pouvoir comparer les intervalles de temps au sein d'une même séquence temporelle et entre différentes séquences temporelles, nous nous limitons dans ce travail aux propriétés qui sont temporellement fonctionnelles (voir la définition 5).

Definition 5 (Propriété temporellement fonctionnelles) *Une propriété p est temporellement fonctionnelle si, pour chaque entité, il n'existe pas une paire d'intervalles se chevauchant dans la séquence temporelle correspondante. Nous notons $\mathcal{FP}$ l'ensemble des propriétés temporellement fonctionnelles. Une propriété p est dans $\mathcal{FP}$ ssi :*

$$\forall x \in \mathcal{I},\ \forall y_1, y_2 \in \mathcal{I} \cup \mathcal{L},\ \forall I_1, I_2 \in \mathcal{T},$$

$$< x, p, y_1, I_1 > \wedge < x, p, y_2, I_2 > \wedge DA(I_1, I_2)$$

Par exemple, `headOfState` est temporellement fonctionnel, puisqu'un président ne peut pas être `headOfState` de deux pays différents dans des intervalles de temps qui se chevauchent.

Dans notre approche, pour une entité donnée, nous générons les séquences temporelles pour toutes ses propriétés temporellement fonctionnelles dans TKG. L'objectif est de générer des contraintes temporelles en comparant les intervalles au sein de la séquence temporelle (intra-séquence) et entre ses séquences temporelles (inter-séquence). Pour éviter la génération de contraintes bruitées et inutiles, nous limitons les comparaisons aux intervalles pertinents qui sont proches dans les séquences, sur la base des définitions suivantes :

Definition 6 (Comparaisons intra-séquence pertinentes) *Pour une séquence temporelle donnée S, les comparaisons intra-séquence pertinentes sont l'ensemble des paires d'intervalles consécutifs.*

Par exemple, dans la figure 2, il y a quatre comparaisons intra-séquence pertinentes : `Meets(3,4)`, `Meets(5,6)`, `Meets(6,7)`, et `Meets(8,9)`. Ces informations sont ensuite stocké dans la matrice $M_{\triangleleft}$.

Definition 7 (Comparaisons pertinentes entre séquences) *Pour une paire donnée de séquences temporelles S et S' des propriétés temporellement fonctionnelles P et P' respectivement, deux intervalles I de S et I' de S' sont considérés comme pertinents pour la comparaison si :*

$$\begin{aligned}
&(I \cap_t I' \neq \emptyset)\\
&\quad \vee\ (I.s < I'.e)\\
&\qquad \wedge(\nexists I'' \in S \setminus \{I\}, (I''.s \geq I.e \wedge I''.s \leq I'.s)\\
&\qquad \wedge(\nexists I'' \in S' \setminus \{I'\}, (I''.e \leq I'.e \wedge I''.e \geq I.s)\\
&\quad \vee\ (I.s > I'.e)\\
&\qquad \wedge(\nexists I'' \in S \setminus \{I\}, (I''.e \geq I'.s \wedge I''.e \leq I.s)\\
&\qquad \wedge(\nexists I'' \in S' \setminus \{I'\}, (I''.s \leq I.s \wedge I''.s \geq I'.e),
\end{aligned}$$

Nous désignons l'ensemble des inter-comparaisons pertinentes entre S et S' par $\Omega(S, S')$.

Dans l'exemple de la figure 2, les comparaisons inter-séquences pertinentes sont illustrées par des flèches. Elles peuvent également être représentées dans une matrice $M_{\triangleright}$ qui peut être utilisée pour indiquer le nombre de comparaisons inter-séquences pertinentes qui remplissent chaque axiome, comme présenté dans le tableau 2. Par exemple, les comparaisons `Starts(5,2)` et `Finishes(8,3)` représentent respectivement les seules comparaisons de début et de fin dans $M_{\triangleright}$.

Relation	o($S_1.I$, $S_2.I$)	o($S_2.I$, $S_1.I$)
Meets	4	0

TABLE 1 – Matrice $M_{\triangleright}$ pour les séquences S_1 et S_2

Relation	o($S_1.I$, $S_2.I$)	o($S_2.I$, $S_1.I$)
Before	2	0
Equals	0	0
Meets	0	0
Overlaps	0	1
During	3	0
Starts	1	0
Finishes	1	0

TABLE 2 – Matrice $M_{\triangleright}$ pour les séquences S_1 et S_2

4.2 Récupération simple des contraintes de temps

Sur la base du nombre d'intervalles dans chaque séquence temporelle, l'algorithme fournit les relations d'Allen o tels que : $\forall I$ d'une séquence S_1, $\exists\ I'$ de la séquence S_2, tel que o($S_1.I$, $S_2.I'$) est satisfait et que $(I, I') \in \Omega(S_1, S_2)$.Nous notons que dans notre méthode, la relation $equal$ est assoupli en une contrainte non symétrique que nous dénotons $subsumes$. Dans l'exemple du tableau 2, il n'y a pas d'axiome d'Allen généralisable pour les deux séquences S_1 et S_2.

4.3 Recherche de contraintes temporelles complexes

Afin d'obtenir des contraintes de temps plus expressives, notre algorithme procède en combinant plusieurs axiomes d'Allen. L'algorithme obtient un ensemble de contraintes temporelles complexes de différents types. D'abord les contraintes complexes qui sont basées sur les relations d'Allen exprimant la disjonction temporelle (DA) et celles qui sont basées sur les relations d'Allen exprimant une certaine intersection (IA). En outre, les contraintes obtenues peuvent être représentées dans un arbre d'ordonnancement, dans lequel les contraintes sont organisées grâce à la relation `isMorePrecise`. Comme dans les relations d'Allen, certaines de nos contraintes complexes sont symétriques, comme `NAND`, alors que `Sequence Meets` ne l'est pas. La figure 3 présente l'arbre d'ordonnancement de toutes les contraintes complexes que nous avons trouvées.

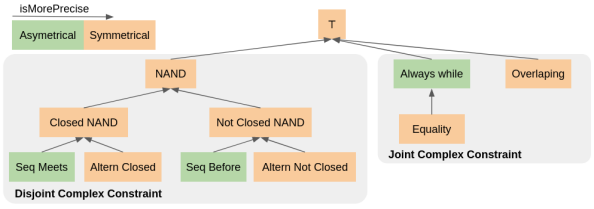

FIGURE 3 – Schema des différentes relation complexes.

Dans ce qui suit, nous définissons formellement les contraintes qui peuvent être trouvées par notre algorithme.

4.3.1 Contrainte NAND.

Pour deux séquences temporelles S et S', et leur ensemble correspondant d'intercomparaisons pertinentes $\Omega(S, S')$, une contrainte NAND exprime que pour chaque intercomparaison pertinente (i, i') il n'y a pas d'axiome intersecté qui soit satisfait.

Definition 8 (Contrainte NAND) *Considérons IA l'ensemble des axiomes intersectés (Définition 1), les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_{\lhd}$ de S et S' remplit la contrainte NAND si :*

$$(\sum_{a \in IA} M_{\rhd}[a][o(P, P')]) = 0$$

4.3.2 Contrainte NAND fermée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte NAND fermée exprime qu'aucune interruption n'apparaît entre le premier et le dernier quadruplet, quelle que soit la séquence temporelle.

Definition 9 *(Contrainte NAND fermée.) Considérons les deux séquences temporelles S et S' des propriétés P et P' respectivement, la matrice des inter-comparaisons $M_{\rhd}$ de S et S', et la matrice des intra-comparaisons $M_{\lhd}$ de S et S' remplissent la contrainte NAND fermée si :*

$$M_{\rhd}[Meets][o(P, P')] + M_{\rhd}[Meets][o(P', P)] +$$

$$M_{\lhd}[Meets][P] + M_{\lhd}[Meets][P'] = |S| + |S'| - 1$$

4.3.3 Contrainte d'alternance fermée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte d'*alternance fermée* exprime qu'après l'apparition d'un quadruplet d'une séquence temporelle, un quadruplet de l'autre séquence temporelle se produira (ou rien si à la fin de la séquence temporelle).

Definition 10 *(Contrainte d'alternance fermée.) Considérons les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_{\rhd}$ de S et S' satisfait la contrainte d'*alternance fermée *si :*

$$M_{\rhd}[Meets][o(P, P')] + M_{\rhd}[Meets][o(P', P)]$$

$$= |S| + |S'| - 1$$

4.3.4 Contrainte de séquence rencontrée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *Sequence Meets* exprime que le dernier quadruplet de S rencontre le premier quadruplet de S'.

Definition 11 *(Contrainte de séquence rencontrée.) Considérons AA l'ensemble des axiomes conjoints (Définition 1), les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_{\rhd}$ de S et S' remplit la contrainte de séquence rencontrée si :*

$$M_{\rhd}[meets][o(P, P')] = 1 \wedge$$

$$(\sum_{a \in AA} M_{\rhd}[a][o(P, P')] + M_{\rhd}[a][o(P', P)]) = 1$$

4.3.5 Contrainte NAND non fermée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *NAND non fermée* exprime qu'un espace apparaît toujours entre les intervalles (inter ou intra séquence temporelle).

Definition 12 *(Contrainte NAND non fermée. Considérons IA l'ensemble des axiomes intersectés (Définition 1), les deux séquences temporelles S et S' des propriétés P et P' respectivement, la matrice des inter-comparaisons $M_{\rhd}$ de S et S', et la matrice des intra-comparaisons $M_{\lhd}$ de S et S' remplit la contrainte NAND non fermée si :*

$$M_{\lhd}[meets][P] + M_{\lhd}[meets][P'] = 0 \wedge$$

$$(\sum_{a \in AA/\{before\}} M_{\rhd}[a][o(P, P')] + M_{\rhd}[a][o(P', P)]) = 0$$

4.3.6 Contrainte d'alternance non fermée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *Alternance non fermée* exprime qu'après l'apparition d'un quadruplet d'une séquence temporelle, un quadruplet de l'autre séquence temporelle se produira après un intervalle (ou rien si à la fin de la séquence temporelle).

Definition 13 *(Contrainte d'alternance non fermée.) Considérons les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_{\rhd}$ de S et S' satisfait la contrainte d'alternance non fermée si :*

$$M_{\rhd}[before][o(P, P')] +$$

$$M_{\rhd}[before][o(P', P)] = |S| + |S'| - 1$$

4.3.7 Contrainte de séquence Before.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *séquence Before* exprime que le dernier quadruplet de S se produit avant tous les autres quadruplets de S'.

Definition 14 *(Contrainte de séquence Before.) Considérons IA l'ensemble des axiomes intersectés (Définition 1), les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_{\rhd}$ de S et S' remplit la contrainte* séquence Before *ssi :*

$$M_{\rhd}[before][o(P, P')] = 1 \wedge$$

$$(\sum_{a \in AA} M_{\rhd}[a][o(P, P')] + M_{\rhd}[a][o(P', P)]) = 1$$

4.3.8 Contrainte d'apparition simultanée.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *d'apparition simultanée* exprime que tous les quadruplets d'une séquence temporelle partagent une intersection avec un autre quadruplet de l'autre séquence temporelle qui est égale à son intervalle temporel (c'est à dire que $q.I \cap_T q'.I = q.I$).

Definition 15 *(Contrainte d'apparition simultanée.) Etant donné la paire de séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_\triangleright$ de S et S' remplit la contrainte d'apparition simultanée si :*

$$M_\triangleright[equals][o(P,P')] + M_\triangleright[during][o(P,P')] +$$

$$M_\triangleright[starts][o(P,P')] + M_\triangleright[finishes][o(P,P')] = |S|$$

4.3.9 Contrainte d'égalité.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte d'*égalité* exprime que chaque quadruplet d'une séquence temporelle a un quadruplet dans l'autre séquence temporelle qui a le même intervalle.

Definition 16 *(Contrainte d'égalité.) Considérons les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_\triangleright$ de S et S' satisfait la contrainte d'égalité si :*

$$M_\triangleright[equality][o(P,P')] +$$

$$M_\triangleright[equality][o(P',P)] = |S| + |S'|$$

4.3.10 Contrainte de chevauchement.

Pour deux séquences temporelles S et S', et leur ensemble correspondant de comparaisons pertinentes $\Omega(S, S')$, une contrainte *chevauchement* exprime que chaque quadruplet chevauche un quadruplet de l'autre séquence temporelle (à l'exception du quadruplet qui commence le plus tard).

Definition 17 *(Contrainte de chevauchement.) Considérons IA l'ensemble des axiomes intersectés (Définition 1), les deux séquences temporelles S et S' des propriétés P et P' respectivement, et la matrice d'inter-comparaisons $M_\triangleright$ de S et S' remplit la contrainte de chevauchement si :*

$$M_\triangleright[overlapping][o(P,P')] +$$

$$M_\triangleright[overlapping][o(P',P)] = |S| + |S'| - 1$$

4.4 Généralisation des contraintes temporelles

Dans les sections précédentes, nous avons décrit comment des contraintes temporelles simples ou complexes peuvent être récupérées à partir de deux séquences temporelles pour une seule entité $e \in C$. Dans cette section, nous présentons notre approche pour généraliser les contraintes découvertes parmi les entités de la classe C.

Pour évaluer si une contrainte peut être généralisée pour une classe C, nous introduisons deux mesures : le *taux d'erreur* et le *taux de généralisation*. La première permet de prendre en compte l'imperfection des données dans un TKG lorsqu'une entité est décrite avec des informations temporelles erronées, ou de surmonter la présence d'entités aberrantes qui ne suivent pas un comportement temporel similaire à celui d'autres entités dans C (voir la définition 18). Cette dernière méthode permet de filtrer les contraintes qui ne sont pas partagées par un pourcentage minimum d'entités dans C (voir la définition 19). Ensuite, étant donné un seuil d'erreur err et un seuil de généralisation gen, nous sélectionnons toutes les contraintes qui peuvent être généralisées, c'est-à-dire toutes les contraintes ayant un taux d'erreur supérieur à err et un taux de généralisation inférieur à gen (voir la définition 20). Enfin, parmi les contraintes généralisées restantes, nous ne conservons que celles dont la relation complexe est la plus précise (comme décrit précédemment dans la section 4.3) afin de filtrer les contraintes temporelles redondantes.

Definition 18 (Taux d'erreur) *Etant donné la contrainte temporelle TC entre les propriétés P et P', l'ensemble des entités $E_{P,P'}$ de la classe C qui sont décrites par les deux propriétés, et le sous-ensemble d'entités $X_{P,P'} \subseteq E_{P,P'}$ où TC a été réfuté. Nous définissons le taux d'erreur comme suit :*

$$ErrorRate(TC) = \frac{|X_{P,P'}|}{|E_{P,P'}|}$$

Definition 19 (Taux de généralisation) *Etant donné la contrainte temporelle TC entre les propriétés P et P', l'ensemble des entités E de la classe C, et l'ensemble des entités $E_{P,P'} \subseteq E$ qui sont décrites par les deux propriétés. Nous définissons le taux de généralisation comme suit :*

$$GeneRate(TC) = \frac{|E_{P,P'}|}{|E|}$$

Definition 20 (Contraintes temporelles généralisées) *Étant donné un seuil d'erreur $err \in [0,1]$ et un seuil de généralisation $gen \in [0,1]$, une contrainte de temps TC peut être généralisée ssi : $ErrorRate(TC) \leq err \wedge GeneRate(TC) \geq gen$.*

4.5 Extension aux propriétés fonctionnelles non temporelles

Restreindre l'approche aux propriétés du TKG qui sont temporellement fonctionnelles peut conduire à ne pas prendre en compte des contraintes pertinentes, en particulier pour certaines propriétés ayant un large éventail de types de valeurs. Par exemple, une personne peut être liée par la propriété `memberOf` à différentes valeurs (par exemple, un groupe musical ou une certain congrès), chacune désignant un type d'assertion différent dans lequel les intervalles de temps sont susceptibles de se croiser dans la séquence temporelle. Ainsi, la probabilité de découvrir des contraintes temporelles pertinentes pouvant être généralisées à toutes les entités de la classe est plus faible. Par

exemple, la propriété `memberOf` peut être spécialisée par valeur en `memberOf-USACongress` afin de découvrir des contraintes plus pertinentes entre la séquence temporelle spécialisée par valeur pour cette propriété et d'autres séquences temporelles (voir la définition 21).
Par conséquent, l'extension proposée dans cette section vise à améliorer la portée et la précision de notre approche, en spécialisant toutes les propriétés qui ont une entité comme valeur. Ces propriétés spécialisées sont ensuite utilisées pour générer les contraintes de la même manière que les propriétés normales. Pour les grands TKG, le processus de spécialisation des valeurs peut être limité aux propriétés qui ont comme valeurs des entités qui sont communément partagées entre plusieurs entités.

Definition 21 (Valeur Séquence de temps spécialisée) *La séquence temporelle spécialisée d'une entité x d'une propriété p et pour une valeur v dans$\mathcal{I}$ est l'ensemble ordonné S d'un ensemble de quadruplets $\{q_1, \ldots, q_n\}$, sous la forme de $< x, p, v, I_k >$ tel que $I_1.start$ est le plus tôt $I_n.start$ est le plus tard.*

5 Validation des faits temporels basée sur des contraintes

Après avoir décrit notre approche de la découverte et de la généralisation des contraintes temporelles, nous présentons dans cette section comment une seule contrainte peut être appliquée pour valider ou réfuter un fait (section 5.1). Nous décrivons ensuite comment toutes les contraintes peuvent être combinées et utilisées pour valider ou réfuter un fait (section 5.2).

5.1 Application d'une contrainte unique

Pour vérifier la validité d'un fait (s, p, o, t) dans un TKG, nous recherchons toutes les contraintes temporelles qui sont pertinentes pour ce fait. Pour qu'une contrainte $tc = o(P_1, P_2)$ soit pertinente pour valider ou réfuter un fait, l'une des propriétés incluses dans une contrainte doit être liée à p : soit P_k = p, soit P_k représente la spécification de la valeur de p, avec $P_k \in P_1, P_2$.
Lorsqu'une contrainte temporelle est pertinente pour un fait, nous récupérons les séquences temporelles de l'entité s pour les relations P_1 et P_2. Ensuite, nous insérons le fait que nous voulons valider dans les séquences temporelles de s pour vérifier son comportement. Nous proposons deux stratégies d'insertion, illustrées dans la figure 4 : une **insertion naïve** et une **suppression d'insertion**. Dans la première stratégie, nous ajoutons l'intervalle t du fait à la séquence temporelle de s, sans tenir compte du fait que cette insertion rompt la fonctionnalité temporelle de la propriété. Dans la stratégie de suppression de l'insertion, nous supprimons tous les intervalles de la séquence qui partagent une partie de leur ligne temporelle avec l'intervalle t.
Enfin, après l'insertion de l'intervalle du fait en utilisant l'une des deux stratégies, nous vérifions si les deux séquences temporelles sont toujours temporellement fonctionnelles et non vides. Si les deux conditions sont remplies,

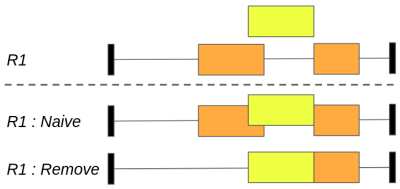

FIGURE 4 – Stratégies d'insertion dans la séquence temporelle $R1$ pour l'intervalle t du fait que nous cherchons à valider (représenté en jaune).

nous vérifions si les séquences temporelles respectent toujours la contrainte temporelle tc (ce qui indique que le fait est temporellement valide), ou si tc est maintenant violé (ce qui indique que le fait est réfuté).

5.2 Application globale des contraintes

Dans la section précédente, nous avons décrit comment la validité temporelle d'un fait peut être vérifiée par rapport à une seule contrainte temporelle. Cette section décrit comment un ensemble de contraintes temporelles peut être combiné et utilisé pour valider ou réfuter un fait. Nous proposons deux stratégies de combinaison : une *approche symbolique* basée sur un système de vote, et une *approche neuro-symbolique* qui tire parti des techniques d'apprentissage automatique pour apprendre quelles contraintes temporelles sont les plus appropriées pour valider temporellement un fait.

5.2.1 Symbolique (Système de vote)

Dans cette stratégie de combinaison, nous vérifions la validité temporelle d'un fait par rapport à toutes ses contraintes temporelles pertinentes. En utilisant un système de vote simple, nous comptons le nombre de contraintes que le fait respecte et le nombre de contraintes que le fait viole. Si le pourcentage de contraintes respectées est supérieur à un seuil minimum, alors notre approche considère ce fait comme temporellement valide. Cette stratégie a l'avantage de rendre la décision résultante complètement explicable, c'est-à-dire que l'on peut retrouver toutes les contraintes temporelles qui ont été utilisées pour justifier la validation ou la réfutation d'un fait.

5.2.2 Neuro-Symbolique

Dans cette stratégie de combinaison, nous vérifions la validité temporelle d'un fait par rapport à toutes les contraintes temporelles disponibles. Ensuite, pour chaque contrainte temporelle tc, nous associons un nombre pour représenter tous les comportements possibles : 0 si la tc est respectée par le fait ; 1 si la tc est violée ; et différentes valeurs pour indiquer si la tc n'est pas pertinente pour le fait, si la fonctionnalité temporelle de la séquence de la propriété est rompue, ou si l'une des séquences temporelles est vide. Il en résulte une matrice $n * m$, où n est la taille de l'ensemble des contraintes temporelles et m le nombre de faits à vérifier, ainsi qu'un vecteur de vérité terrain de dimension n qui peut être utilisé pour former et tester le modèle d'apprentissage automatique.

Classe	# Entités	# Quadruplets
Pays (Q6256)	205	183 249
Groupe de musique (Q215380)	55 507	131 476
Politicien (Q82955)	658 445	2 085 232

TABLE 3 – Description des trois ensembles de données

6 Evaluation expérimentale

Dans cette section, nous présentons l'évaluation expérimentale de notre cadre sur une série d'ensembles de données. Toutes les expériences sont réalisées sur un processeur "Intel® Xeon® E5-2630 v4" avec 10 cœurs et 128 Go de RAM. Le code source et les jeux de données sont disponible sur ce dépot github. [3]

6.1 Jeux de données

Nous testons notre approche sur trois ensembles de données extraits de Wikidata [10], représentant tous les faits liés aux entités de type Pays (Q6256), Groupe musical (Q215380) et Homme politique (Q82955). Le tableau 3 indique le nombre d'entités pour chaque classe et le nombre total de faits temporels (quadruplets) les décrivant.

Les ensembles de données ont été divisés selon un ratio de 80%, 10%, 10% pour l'ensemble de formation, de validation et de test. L'échantillonnage négatif a été effectué en changeant de manière aléatoire la partie temporelle de chaque fait autour de la durée de vie de l'entité. Ainsi, pour une entité ayant existé entre 1900 et 2000, la valeur aléatoire ne prendra sa valeur qu'entre 1850 et 2050 afin de créer des faits raisonnablement faux. L'ensemble d'entraînement échantillonné non négatif sera utilisé pour découvrir la contrainte temporelle, tandis que l'ensemble augmenté servira à l'entraînement de l'algorithme d'apprentissage automatique.

6.2 Step by step tuning

Pour évaluer les différents hyper-paramètres de notre approche, nous allons procéder étape par étape en réglant d'abord la stratégie pour la décision finale. Ensuite, le type de contrainte temporelle autorisé (le type par défaut sera uniquement avec des relations), suivi de la stratégie d'insertion (la stratégie par défaut sera la stratégie naïve). Puis la stratégie d'insertion (la stratégie par défaut sera la stratégie naïve), et enfin le réglage du seuil d'erreur en même temps que celui de la généralisation (la valeur par défaut sera $ET = 5\%$ et $GT = 5\%$). Chaque expérience sera ensuite évaluée selon deux métriques : **Accuracy** (Acc.) et la **Coverage** (cov.) réalisée. Nous noterons également le temps d'exécution (RT), de la découverte des contraintes à l'application, de chaque hyperparamètre.

6.2.1 Stratégies de combinaison des contraintes

La première expérience consiste à évaluer quelle stratégie, utilisée pour combiner toutes les contraintes temporelles de validation ou de réfutation d'un fait, permet d'obtenir de meilleures performances. Le tableau 4 présente les résultats de cette expérience sur les trois classes. Il montre que la stratégie neuro-symbolique peut valider des faits avec une précision légèrement supérieure à celle de la stratégie symbolique pour toutes les classes testées. Par conséquent, nous choisirons la stratégie de combinaison neuro-symbolique pour le reste des expériences, malgré l'augmentation significative du temps d'exécution.

3. https ://github.com/SoulardThibaut/TemporalConstraints

Classe	Deci. Type	Acc.	Cov.	RT
Country	Symbolic	79.5	9.4	**2m 30s**
	Neuro-Symb.	**80.4**	9.4	52m
Groupe de musique	Symb.	64.0	37.5	**2m 50s**
	Neuro-Symb.	**64.3**	37.5	5m 50s
Politicien	Symb.	61.6	44.0	**44m**
	Neuro-Symb.	**62.3**	44.0	1h 30m

TABLE 4 – Comparaison des stratégies de combinaison de contraintes

Classe	Const. Type	Acc.	Cov.	RT
Pays	R	80.4	9.4	**52m**
	R & RxV	**90.8**	**14.1**	2h 35m
Groupe de musique	R	64.2	37.5	5m 50s
	R & RxV	64.2	37.5	5m 50s
Politicien	R	61.6	44.0	1h 30m
	R & RxV	61.6	44.0	1h 30m

TABLE 5 – Comparaison des spécialisations

6.2.2 Contraintes spécialisation des valeurs

La deuxième expérience consiste à évaluer si l'ajout de séquences temporelles spécialisées (RxV) permet d'obtenir de meilleures performances par rapport à l'utilisation de séquences temporelles normales (R). Le tableau 5 montre les avantages de l'extension de notre approche pour la classe Politicien (Q6256), où le pourcentage de couverture qui peut être fait est augmenté de près de 50% (de 9,4 à 14,1) et le nombre d'évaluations exactes est augmenté de 13% (de 80,4 à 90,8). Par conséquent, nous appliquons cette extension pour les expériences restantes, malgré l'absence d'amélioration pour les deux classes restantes, car elle ne présente pas d'inconvénients.

6.2.3 Stratégie d'insertion

Cette expérience consiste à comparer les deux stratégies d'insertion présentées à la section 5.1. Le tableau 6 montre que la stratégie de suppression d'insertion réduit légèrement la précision de notre approche, mais qu'en contrepartie, elle augmente légèrement le pourcentage de couverture possible. Pour les expériences suivantes, nous avons utilisé la stratégie de suppression d'insertion car elle permet d'évaluer le plus grand nombre de faits.

Classe	Insert Type	Acc.	Cov.	RT
Country	Naive	**90.8**	14.1	**2h 35m**
	Remove	88.5	**14.4**	2h 54m
Groupe de musique	Naive	64.2	37.5	5m 50s
	Remove	64.2	37.5	5m 50s
Politicien	Naive	**62.3**	44.0	**1h 30m**
	Remove	62.1	**46.9**	1h 32m

TABLE 6 – Comparaison des stratégies d'insertions

Classe	gen	err	Acc.	Cov.	RT
Country	2	10	-	-	-
	2	5	**88.6**	16	8h 50m
	5	10	87.9	**17.2**	4h 30m
	5	5	88.5	14.4	**2h 54m**
Groupe de musique	2	10	**64.6**	**38.2**	6m 6s
	2	5	**64.6**	**38.2**	5m 54s
	5	10	64.2	37.5	**5m 51s**
	5	5	64.2	37.5	**5m 51s**
Politicien	2	10	63.4	**51.9**	1h 35m
	2	5	62.1	49.9	**1h 32m**
	5	10	**63.5**	48.9	1h 34m
	5	5	62.1	46.9	**1h 32m**

TABLE 7 – Comparaison des seuils d'erreur et de généralisation

6.2.4 Seuils d'erreur et de généralisation

Le tableau 7 présente les résultats pour différents seuils d'erreur err et de généralisation gen. Nous pouvons constater qu'un err plus élevé et un gen plus faible permettent à notre approche d'évaluer un plus grand nombre de faits, sans impact significatif sur la précision. Cependant, cela se fait au prix d'un temps d'exécution plus élevé car l'approche dispose d'un plus grand nombre de contraintes temporelles à utiliser pour valider ou réfuter un fait. Le nombre élevé de contraintes peut entraîner des problèmes d'évolutivité en termes de mémoire, par exemple, la première ligne où la présence d'environ 24.000 de contraintes temporelles (c'est-à-dire de caractéristiques) crée un problème pour le *Decision Tree Classifier*.

7 Conclusion

Dans cet article, nous avons présenté une nouvelle approche pour évaluer la validité temporelle des faits dans un graphe de connaissances. L'approche utilise et étend l'algèbre d'intervalles d'Allen pour découvrir les contraintes temporelles du graphe de connaissances. Pour l'évaluation, nous avons procédé à une mise au point étape par étape afin d'évaluer et d'expliquer l'impact de chaque stratégie proposée dans ce travail. Grâce à ces expériences, nous avons montré que notre approche peut valider ou réfuter un fait temporel avec une grande précision (jusqu'à 90.8%), malgré une couverture relativement faible (couverture maximale de 51.9%).

À l'avenir, nous nous efforcerons de résoudre les différents problèmes soulevés par les expériences. Nous prévoyons tout d'abord de réduire le nombre de caractéristiques qui ont un impact important sur les performances de l'approche neuro-symbolique, en éliminant les contraintes temporelles qui sont moins importantes. Ensuite, comme nous n'avons considéré que des ensembles de données extraits d'une seule source (Wikidata), nous n'avons pas pu évaluer si l'ensemble des contraintes temporelles découvertes dans un graphe peut être transféré et utilisé pour valider ou réfuter des faits temporels dans un autre graphe avec une précision et une couverture élevées. C'est pourquoi nous souhaitons tester la transférabilité de ces contraintes temporelles sur plusieurs autres KG temporels, tels que YAGO [8].

Références

[1] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. *Data Profiling*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2018.

[2] James F Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11) :832–843, 1983.

[3] Borui Cai, Yong Xiang, Longxiang Gao, Heng Zhang, Yunfeng Li, and Jianxin Li. Temporal knowledge graph completion : A survey. In *International Joint Conference on Artificial Intelligence*, 2022.

[4] Melisachew Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. Marrying uncertainty and time in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[5] Cristian Consonni, Paolo Sottovia, Alberto Montresor, and Yannis Velegrakis. Discovering order dependencies through order compatibility. In Melanie Herschel, Helena Galhardas, Berthold Reinwald, Irini Fundulaki, Carsten Binnig, and Zoi Kaoudi, editors, *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*, pages 409–420. OpenProceedings.org, 2019.

[6] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. *arXiv preprint arXiv :1809.03202*, 2018.

[7] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, pages 1771–1776, 2018.

[8] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3 : A knowledge base from multilingual wikipedias. In *CIDR*, 2013.

[9] Vangipuram Radhakrishna, P. V. Kumar, and V. Janaki. A survey on temporal databases and data mining. In *Proceedings of the The International Conference on Engineering & MIS 2015*, ICEMIS '15, New York, NY, USA, 2015. Association for Computing Machinery.

[10] Denny Vrandečić and Markus Krötzsch. Wikidata : A free collaborative knowledgebase. *Commun. ACM*, 57(10) :78–85, September 2014.

[11] Jiapu Wang, Boyue Wang, Meikang Qiu, Shirui Pan, Bo Xiong, Heng Liu, Linhao Luo, Tengfei Liu, Yongli Hu, Baocai Yin, and Wen Gao. A survey on temporal knowledge graph completion : Taxonomy, progress, and prospects, 2023.

[12] Renjie Xiao, Zijing Tan, Haojin Wang, and Shuai Ma. Fast approximate denial constraint discovery. *Proc. VLDB Endow.*, 16(2) :269–281, oct 2022.

[13] Jiasheng Zhang, Shuang Liang, Yongpan Sheng, and Jie Shao. Temporal knowledge graph representation learning with local and global evolutions. *Knowledge-Based Systems*, 251 :109234, 2022.

[14] Jiasheng Zhang, Yongpan Sheng, Zheng Wang, and Jie Shao. Tkgframe : a two-phase framework for temporal-aware knowledge graph completion. In *Web and Big Data : 4th International Joint Conference, APWeb-WAIM 2020, Tianjin, China, September 18-20, 2020, Proceedings, Part I 4*, pages 196–211. Springer, 2020.