Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles
Avignon, France, 27 juin au 1er juillet 2022

Construction de Graphes de Connaissances à partir de Textes avec une I.A. Centrée-Utilisateur

Hugo Ayats
Univ Rennes 1, CNRS, IRISA
Campus de Beaulieu, 35042 Rennes, FRANCE
prenom.nom@irisa.fr

RÉSUMÉ

Avec l'essor du Web sémantique au cours des deux dernières décennies est apparu un besoin en outils permettant de construire des graphes de connaissances de bonne qualité. Cet article présente mon travail de thèse, qui est la conception d'une méthode explicable et centrée-utilisateur pour la production semi-automatisée de graphes de connaissances à partir de textes spécifiques à un domaine. Ce système se présente initialement comme une interface d'édition guidée de RDF. Puis, se basant sur les actions de l'utilisateur, un système de suggestion de triplets se met en place. Enfin, à travers des interactions avec l'utilisateur, le système automatise progressivement le processus. Après avoir présenté le *workflow* du système et détaillé les unités qui le compose – une unité de prétraitement, une unité interactive et une unité automatisée - cet article documente les aspects de ce *workflow* déjà implémentés, ainsi que les résultats de leur évaluation.

ABSTRACT

Knowledge Graph Construction from Texts with an User-Centric A.I.

With the rise of the Semantic Web over the last two decades, there has been a need for tools to build good quality knowledge graphs. This paper presents my thesis work, which is the design of an explainable, user-centered method for the semi-automated production of knowledge graphs from domain-specific texts. This system is initially presented as a guided RDF editing interface. Then, based on the user's actions, a triplet suggestion system is implemented. Finally, through interactions with the user, the system gradually automates the process. After presenting the *workflow* of the system, and detailing the units that compose it - a pre-processing unit, an interactive unit and an automated unit - this article details the aspects of this *workflow* already implemented, as well as the results of their evaluation.

MOTS-CLÉS : Web sémantique, Graphe de Connaissances, I.A. centrée-utilisateur, explicabilité, Graph-FCA.

KEYWORDS: Semantic Web, Knowledge Graph, user-centric A.I., explainability, Graph-FCA.

1 Introduction

Durant les deux dernières décennies, le domaine du Web Sémantique est devenu partie intégrante du World Wide Web, avec à la fois un champ de recherche académique actif et de nombreuses applications industrielles. Aujourd'hui, le Linked Open Data Cloud regroupe plus de 1200 graphes de connaissances (KGs). Des KGs tels que YAGO (Pellissier Tanon *et al.*, 2020) ou Wikidata (Vrandečić

& Krötzsch, 2014) regroupent plusieurs dizaines de millions de faits, et de nombreux travaux présentent comment requêter, éditer, unifier ou compléter ces KGs (Hitzler, 2021). Différentes approches sont utilisées pour construire ces KGs. YAGO découle d'une collecte automatique de faits depuis des pages Web structurées, alors que Wikidata est mis à jour et complété à la main par sa communauté.

On pourrait penser que les récents progrès en IA, induits par l'explosion du *deep learning*, auraient profité au domaine de la construction automatique de KGs. En effet, ces progrès ont touché le domaine du Traitement Automatique des Langues (TAL) et de l'extraction d'information, notamment par exemple avec l'apparition de modèles de langue préentrainés tels que BERT (Devlin *et al.*, 2019) et ses dérivés. Cependant, les scores obtenus par ces systèmes sont encore trop bas pour permettre une automatisation totale de la construction de KGs à partir de textes. L'idée de ma thèse est que des approches centrées-utilisateur, mettant en avant l'interactivité et l'explicabilité, et rendant ainsi les systèmes plus intelligibles et plus fiables, pourrait être une solution.

Cet article présente mon sujet de thèse, dont l'objectif est le développement d'une IA centrée-utilisateur pour la construction de KGs à partir de textes. Ce système, à travers des interactions avec l'utilisateur, automatise progressivement le processus, en proposant initialement une interface pour la création manuelle de KG, puis en formulant des suggestions couplées avec des explications en se basant sur les actions précédentes de l'utilisateur, pour enfin peupler le graphe de façon de plus en plus automatisée.

Afin de réduire la taille du vocabulaire et d'avoir des cas d'usage pratiques, nous visons en priorité une utilisation sur des textes spécifiques à un domaine, par exemple, un juriste traitant et archivant des rapports de procès. Dans cet exemple, l'utilisateur se constituerait progressivement un graphe de connaissances récapitulant les acteurs des différents procès et leurs rôles, et ceci à faible coût, la plus grosse part du travail se faisant automatiquement après un certain temps. Le KG ainsi obtenu pourra être exploité afin de retrouver l'historique d'un individu particulier ou à des fins statistiques.

Dans la suite de cet article, la section 2 présente l'état de l'art. La section 3 détaille la conception du système et de ses sous-systèmes. Pour finir, la section 4 présente les premiers résultats obtenus avec les éléments du système déjà implémentés.

2 État de l'art

Au cours de ces vingt-cinq dernières années, de nombreuses méthodes ont été mises en œuvre pour la construction et l'édition de graphes de connaissances (KGs). L'approche la plus évidente est l'approche manuelle, consistant à éditer à la main le graphe cible. L'outil d'édition manuelle de KG le plus célèbre est *Protégé* (Musen, 2015), logiciel développé par l'Université de Stanford depuis les années 1990 et spécialisé pour la création et l'édition d'ontologies. Aujourd'hui, certains KGs massifs tel que Wikidata (Vrandečić & Krötzsch, 2014) reposent toujours sur l'édition manuelle par ses utilisateurs, à travers une interface prenant la forme d'un formulaire. D'autres KGs, tels que YAGO (Steiner *et al.*, 2012) ou le Google Knowledge Graph (Pellissier Tanon *et al.*, 2020), reposent sur l'extraction massive de faits à partir de pages Web structurées, telles que les *infobox* de Wikipedia ou les sites de e-commerce. Parmi les outils plus expérimentaux, on peut citer (Maillot *et al.*, 2017) qui propose une interface d'édition interactive de KG se basant sur un moteur de recommandation.

Une tâche centrale pour la contruction automatique de KGs à partir de textes est la tâche d'extraction de relations. Cette tâche consiste, pour une phrase donnée dans laquelle sont identifiés deux entités, à

prédire quelle relation lie ces deux entités. En pratique, cela permet d'extraire automatiquement des faits pouvant servir à peupler des KGs. Si historiquement, des approches symboliques se sont attelées à cette tâche, notamment par des approches à base de règles validées manuellement, aujourd'hui l'état de l'art est constitué d'approches *deep learning*. Initialement, des approches utilisant des réseaux convolutionnels (Nguyen & Grishman, 2015), des réseaux récurents de type LSTM (Tai *et al.*, 2015) ou de la convolution de graphe (Zhang *et al.*, 2018) ont été utilisées. Cependant, elles sont aujourd'hui largement dominées par les approches utilisant des modèles de langue pré-entrainés, basés sur la technologie des *transformers* (Vaswani *et al.*, 2017), tel que BERT (Devlin *et al.*, 2019) et ses dérivés (Yamada *et al.*, 2020; Lyu & Chen, 2021). Cependant, ces approches sont toujours trop peu performantes (F-scores de l'ordre de 0.75 sur le benchmark TACRED (Zhang *et al.*, 2017)) pour permettre une utilisation sans curation manuelle. On peut aussi citer (Martinez-Rodriguez *et al.*, 2018) qui détaille une méthode visant à construire automatiquement un KG à partir de textes en utilisant des techniques issues du domaine de l'*Open Information Extraction*, méthode qui retourne en pratique un KG difficilement exploitable.

Concernant les I.A. centrées-utilisateur, ce domaine relativemment récent a fait l'objet de bien moins de travaux. On peut cependant citer (Shneiderman, 2020), qui théorise la possibilité d'avoir des systèmes à la fois hautement automatisés et fortement contrôlés par l'humain, avec pour objectif qu'ils soient fiables, sûrs et dignes de confiance. En parallèle, les travaux présentés dans (Ferré & Ridoux, 2002) proposent un classifieur pour e-mails basé sur une extension logique de l'analyse des concepts formels (FCA) (Ganter & Wille, 1996) qui fonctionne selon un cycle similaire à celui présenté dans cet article : annotation manuelle, puis suggestions du système, et enfin automatisation.

3 Approche proposée

L'objectif global de cette thèse est la création d'un système centré utilisateur pour la production semi-automatique de KGs à partir de textes en langue naturelle. Plus précisément, ce système aura pour objectif de produire des graphes RDF de haute qualité à partir de textes sans aucune connaissance ni aucun vocabulaire préalable en automatisant progressivement les actions de l'utilisateur. Initialement, le système proposera une interface pour l'édition de graphes RDF. Puis, se basant sur les actions de l'utilisateur, le système recommandera des triplets à ajouter au graphe. Finalement, le système automatisera la validation des suggestions les plus fiables, afin d'obtenir un processus de génération de graphes en grande partie automatisé, l'utilisateur n'étant sollicité que pour les cas ambigus ou inédits.

Cette section présente en détail le *workflow* d'un tel système. Premièrement, une vue globale du système est donnée, avant de détailler les rôles et possibles implémentations de ses sous-systèmes.

3.1 Présentation générale

Tel qu'on peut le voir sur la figure 1, le système peut être divisé en trois unités. La première a un rôle de pré-traitement. Elle prend en entrée le texte brut et, après des traitements TAL et une phase de détection de relations, le traduit dans un modèle syntaxico-sémantique. Ce modèle est alors utilisé par deux unités parallèles. La première, l'unité interactive, est composée d'une IA explicable pour la classification de relations qui vient produire des triplets candidats ainsi que des explications pour ces prédictions. Candidats et explications sont alors fournis à l'utilisateur pour validation. Les triplets

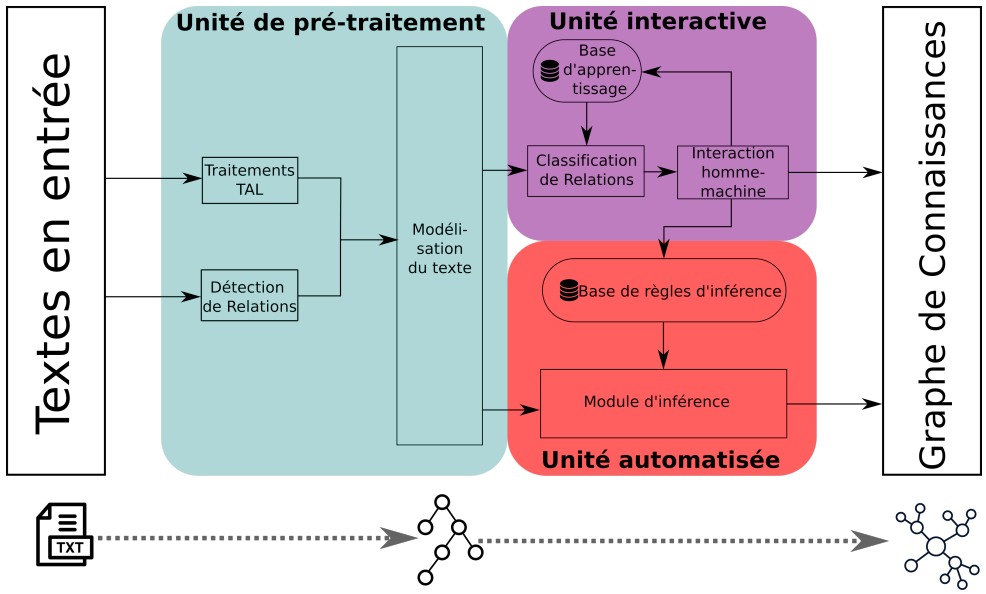

FIGURE 1 – Vue globale du workflow

validés sont alors utilisés pour peupler le graphe de connaissances, et les explications validées sont transformées en règles d'inférence. Ces règles sont alors utilisées par la troisième unité, l'unité automatisée, qui les utilise pour générer des triplets fiables directement à partir de la modélisation du texte.

On peut constater que ce *workflow* contient deux bases : la première est une base d'apprentissage composée d'exemples annotés pour la classification de relations, utilisée par le module de classification de relations, alors que la seconde est constituée de règles d'inférence dérivées des explications issues du module de classification de relations et utilisées par le module d'inférence pour produire des triplets. Afin d'avoir un système indépendant de tout vocabulaire, ces deux bases devront être initiallement vides. Cela a pour conséquence que ce système est incrémental : initialement, le module d'extraction de relation n'est pas capable de produire des prédictions. Le système se comporte donc comme une interface pour l'edition guidée de graphes RDF le temps de venir alimenter cette base. Puis, alors que cette base grossit, le module d'extraction de relations devient capable de produire des prédictions de plus en plus fiables venant aider l'utiliateur. Enfin, la validation de certaines suggestions et explications par l'utilisateur produit des règles d'inférence venant nourrir la base utilisée par l'unité automatisée, automatisant ainsi progressivement le processus.

3.2 Unité de pré-traitement

Comme évoqué précédemment, le rôle principal de ce sous-système est de transformer le texte en langue naturelle d'entrée en un modèle utilisable par les deux autres unités. Cette modélisation doit avoir deux propriétés. Tout d'abord, elle doit regrouper toutes les informations linguistiques contenues dans le texte (entités nommées, catégories morphosyntaxiques, dépendances grammaticales...) qui pourront être utilisées pour l'extraction de triplets. De plus, cette modélisation doit spécifier entre quelles entités il pourrait y avoir une relation traduisible en triplet. Ce second aspect est nécessaire pour deux raisons. Premièrement, cela permet au système de suggérer à l'utilisateur des couples d'entités à transformer en triplets avant que le module d'extraction de relations ait assez d'exemples

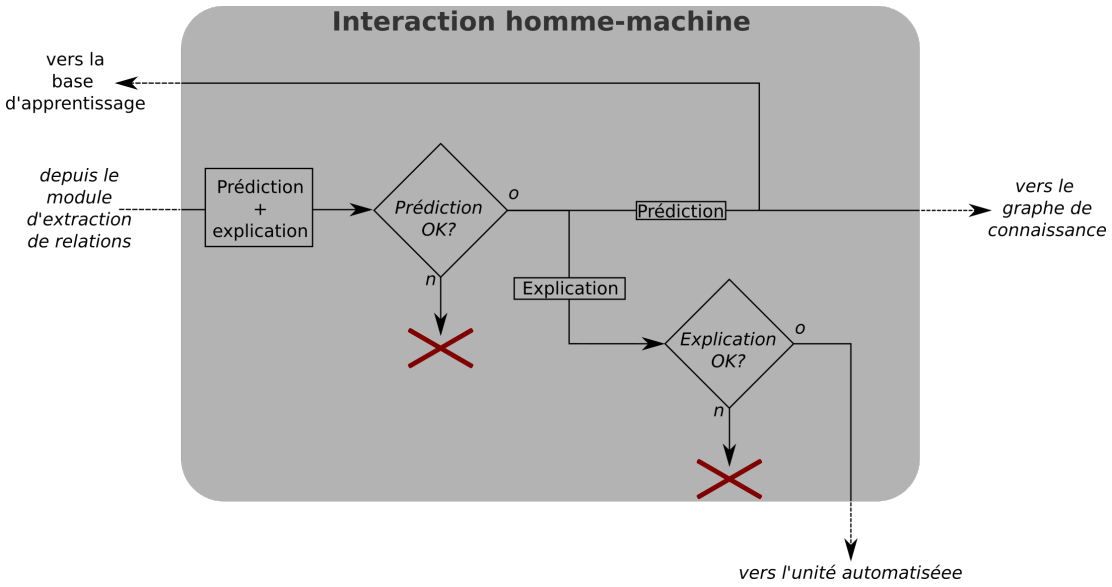

FIGURE 2 – Schéma d'interaction

annotés pour prendre le relais. Ensuite, le module que nous comptons utiliser pour l'extraction de relations est uniquement apte à faire de la classification de relation, c'est à dire à prédire quelle relation lie deux entités en sachant que ces deux entités sont effectivement liées par une relation. Ce résultat est discuté plus en détail dans la section 4.

Par conséquent, cette unité de pré-traitement peut être décomposée en trois modules. Le premier est un ensemble d'outils de TAL, tel que *Stanford CoreNLP* (Manning *et al.*, 2014), capable d'extraire des données linguistiques (*i.e.* entités nommées, catégories morphosyntaxiques, etc.) du texte. Le second est un module de détection de relations, chargé de détecter quelles paires d'entités au sein du texte peuvent faire l'objet d'une relation traduisible en triplet. Enfin, le troisième module transforme les données produites par les deux précédents en un modèle linguistique et sémantique du texte en entrée. On peut remarquer que, dans cette unité de pré-traitement, l'explicabilité n'est pas exigée : on cherche simplement à avoir les outils les plus efficaces, et nous comptons sur l'interactivité présente dans le reste du système pour corriger les omissions et erreurs possibles.

3.3 Unité interactive

Ce sous-système a deux rôles principaux. Tout d'abord, il doit contenir un module explicable de classification de relations capable de travailler sur la modélisation produite par l'unité de pré-traitement. Comme la base d'apprentissage de ce module évolue en s'enrichissant au cours de l'utilisation du système, ce module devra reposer sur une approche paresseuse (dite aussi *lazy learning*), *i.e.* une méthode faisant un apprentissage pour chaque instance, et n'apprenant pas un modèle une fois pour toutes. De plus, ce module d'extraction de relation doit produire des explications facilement compréhensibles par l'utilisateur, et pouvant être transformées en règles d'inférence pour être utilisé par le dernier sous-système.

Le second rôle de ce sous-système sera de fournir une interface complète pour l'édition de graphe RDF, prenant en compte la modélisation et les paires d'entités retournées par l'unité de pré-traitement, mais permettant aussi l'édition et le typage manuel d'entités ou de relations dans le but de rendre l'utilisateur

capable de corriger les erreurs et omissions faites par l'unité de pré-traitement. Cette interface permettra de peupler à la fois le graphe RDF et la base d'apprentissage du module d'extraction de relations. Cette interface devra aussi prendre en compte les suggestions faites par le module d'extraction de relations, ainsi que leurs explications, pour les soumettre pour validation à l'utilisateur. La figure 2 détaille le processus interactif pour les suggestions : d'abord, l'interface suggère un triplet à ajouter au graphe de connaissances, que l'utilisateur peut accepter ou rejeter. Si le triplet est accepté, il est ajouté au graphe de connaissances et à la base d'apprentissage, et l'explication de cette suggestion est proposée à l'utilisateur. Si cette explication satisfait l'utilisateur, elle est transformée en règle d'inférence et transmise à l'unité automatisée.

3.4 Unité automatisée

Ce sous-système, plus simple que les deux autres, est constitué d'uniquement deux éléments : un module d'inférence et sa base de règles d'inférence. Comme expliqué plus tôt, cette base de règles est initialement vide, et par conséquent le module d'inférence est inactif. Au fur et à mesure que l'utilisateur valide des explications fournies par le module d'extraction de relations, ces explications sont transformées en règles d'inférence et ajoutées à la base de règles. Le module d'inférence applique alors ces règles sur la modélisation du texte dans le but d'en extraire automatiquement des triplets à ajouter au graphe de connaissances. L'objectif globale du système est alors de peupler suffisament cette base de règles afin que la plupart des triplets soient extraits automatiquement, ne laissant à l'utilisateur que les cas ambigus ou inédits.

Cela a pour conséquence que les règles produites, et par extension les explications retournées par le module d'extraction de relations, doivent nécessairement être directement appliquables sur la modélisation du texte. Ainsi, ces règles doivent être de la forme $\mathcal{T}(x, y) \leftarrow \mathcal{P}(x, y)$, avec x et y deux variables, $\mathcal{T}(x, y)$ un triplet RDF et $\mathcal{P}(x, y)$ un motif qui peut être directement appliqué sur la modélisation du texte.

4 Premiers résultats

Dans cette section, je présente les premiers résultats obtenus. La section 4.1 présente comment le texte est modélisé sous forme de graphe, et introduit le module *deep learning* de détection de relations utilisé par cette modélisation. La section 4.2 introduit une approche *lazy-learning* et explicable de classification de relations s'appuyant sur une méthode symbolique. Enfin, la section 4.3 présente les résultats préliminaires obtenus avec ces modules. Le travail présenté dans cette section a déjà fait l'objet de deux publications : (Ayats *et al.*, 2021, 2022).

4.1 Modélisation du texte sous forme de graphe

L'idée derrière cette modélisation est que la sémantique d'un texte repose sur deux éléments : sa structure grammaticale d'une part, et la sémantique des termes qu'il emploit d'autre part. Par conséquent, en représentant ces éléments au sein d'un graphe et en raisonnant sur ce graphe, il devrait être possible d'extraire de la connaissance de ce texte. Pour cette modélisation, nous avons fait le choix de modéliser chaque phrase individuellement. Pour des raisons pratiques, cette modélisation a

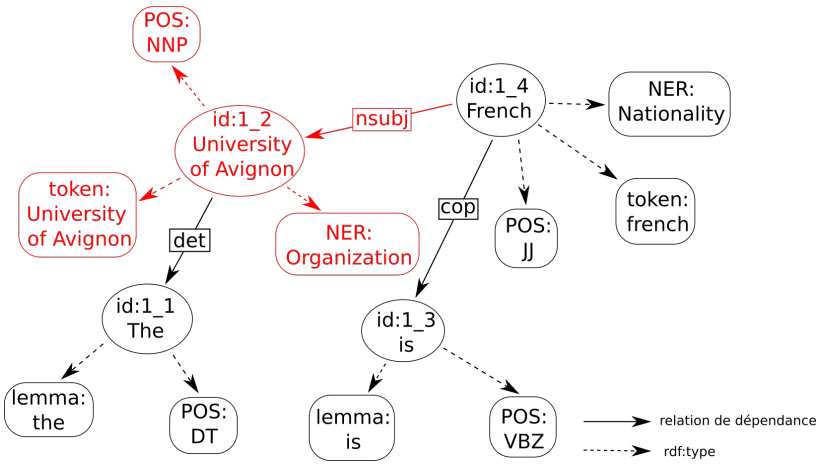

FIGURE 3 – Exemple de modélisation d'une phrase

été faite sous forme de graphes RDF. Cependant, il s'agit bien d'une représentation intermédiaire, et non le graphe de connaissances cible.

Tout d'abord, chaque phrase du texte est analysée par des outils de TAL (un tokeniseur, un lemmatiseur, un détecteur de catégorie morphosyntaxique, un module de reconnaissance d'entités nommées et un parseur grammatical) [1]. Puis, pour chaque token, une entité RDF avec une URI unique est créée, et l'arbre de dépendance de la phrase est reconstruit en liant ces entités avec des relations de dépendance. Ensuite, chaque entité est typé avec sa catégorie morphosyntaxique et son lemme. Afin de traiter correctement les entités nommées, nous avons choisi de représenter les tokens contigus d'une même entité nommée en une seule entité RDF, typée par le type de cette entitée nommée et par la concaténation de ses tokens. La figure 3 présente par exemple la modélisation de la phrase *"The University of Avignon is French"*. Sur ce schéma, on voit que les tokens représentant l'entité *University of Avignon* ont été concaténés en un seul noeud du graphe, noeud qui a été typé par son type en tant qu'entité nommée, sa catégorie morphosyntaxique (*POS tag* en anglais) et par la concaténation de ses tokens. De plus, cette entité a pour parent dans l'arbre de dépendance l'entité *French* par la relation *nsubj*.

De plus, afin d'enrichir la modélisation en permettant la relaxation de types RDF sur certains aspects, nous avons construit une ontologie RDFS par dessus celle-ci. Tout d'abord, quelques catégories morphosyntaxiques ont été hiérarchisées. Par exemple, un nom propre pluriel (NNPS) étant un nom propre (NNP), le triplet *subClassOf(POS :NNPS, POS :NNP)* a été ajouté à l'ontologie. Ensuite, afin d'enrichir sémantiquement cette modélisation, nous avons utilisé la base de données lexicale *Wordnet* (Miller, 1998) pour construire une hiérachie au dessus des lemmes des noms et verbes. Ainsi chaque ensemble de synonymes d'un lemme est utilisé comme superclasse de ce lemme, et chaque hyperonyme de chaque ensemble de synonymes est considéré comme étant une superclasse de cet ensemble de synonymes.

Enfin, afin de compléter cette modélisation, en accord avec le *workflow* présenté, nous avons besoin d'un module de détection de relations, chargé d'indiquer entre quelles paires d'entités de la modélisation il existe une relation susceptible d'être transformée en triplet. Commme évoqué pécédemment, pour ce module nous cherchons la performance brute et n'avons pas besoin d'explicabilité. C'est pourquoi nous nous sommes tournés vers un modèle de langue pré-entrainé.

1. Dans notre cas, nous avons utilisé Stanford CoreNLP (Manning *et al.*, 2014)

LUKE (Yamada *et al.*, 2020) est un modèle de langue pré-entrainé de la famille de BERT (Devlin *et al.*, 2019) qui a la particularité de prendre en compte, en plus de chaque token séparément, la présence d'entités pouvant s'étendre sur plusieurs mots. Cette particularité lui a permis d'établir un nouvel état de l'art sur plusieurs tâches de TAL, dont l'extraction de relations. C'est donc le modèle que nous avons choisi d'adapter pour la détection de relations.

Nous présentons deux configurations de LUKE pour la détection de relations. La première, appellée *luke-base* consiste simplement à réutiliser le modèle entrainé pour l'extraction de relation et modifier la sortie en post-processing afin de fusionner toutes les prédictions prédisant la présence d'une relation en une seule classe. La seconde configuration, appellée *luke-detect* consiste à spécialiser LUKE pour la détection de relations en modifiant les couches de sortie et en faisant un nouveau *fine-tuning*. On peut s'attendre à ce que *luke-detect*, étant plus spécialisé, ait des meilleurs scores que *luke-base* en détection de relations.

4.2 Classification de relations avec les Concepts de Voisins

Comme mis en évidence dans le *workflow*, cette modélisation est utilisée afin de faire de l'extraction de relations. Le principe de cette tâche est de prédire, pour une phrase donnée dans laquelle sont identifiées deux entités (un sujet et un objet) ainsi que le type de ces entités, quelle est la relation liant le sujet et l'objet parmi un jeu de relations donné. Si historiquement cette tâche a été abordée selon diverses approches, aussi bien symboliques que statistiques, aujourd'hui l'état de l'art est dominé, comme dans beaucoup de tâches de TAL, par des méthodes utilisant des modèles de langue pré-entraînés tels que BERT et ses dérivés (Shi & Lin, 2019; Yamada *et al.*, 2020). Ces méthodes sont très performantes, toutefois, leur manque d'interprétabilité les rend inutilisables dans notre contexte.

L'idée derrière la méthode présentée dans cette section est la suivante : pour une phrase donnée, la modélisation présentée précédemment regroupant les informations sémantiques et linguistiques utiles pour comprendre la sémantique de la phrase, la représentation d'un sujet et d'un objet au sein de cette modélisation devrait permettre de déduire la nature de la relation entre ce sujet et cet objet. Plus exactement, pour une phrase de test donnée, si les entités sujet et objet sont liées par une relation donnée, alors la position de cette paire d'entités $(sujet, objet)$ dans la modélisation doit avoir des similarités avec la position des paires d'entités $(sujet', objet')$ de phrases exprimant la même relation. Pour exploiter cette hypothèse, nous utilisons la méthode des Concepts de Voisins.

Le calcul de Concepts de Voisins (Ferré, 2017) est une méthode basée sur Graph-FCA, une extention de l'Analyse de Concepts Formels (*Formal Concept Analysis, FCA*) aux graphes qui est utilisée pour la découverte de similarités dans un graphe. Pour un n-uplet de nœuds du graphe donné, cette méthode calcule les n-uplets les plus similaires parmi un ensemble de n-uplets donnés. Pour ce faire, l'algorithme fait du clustering hiérarchique sur cet ensemble de n-uplets au sein de *concepts*. Chaque concept est défini par son *extension*, l'ensemble des n-uplets qu'il contient, et par son *intension*, qui est le motif de graphe que matchent tous les n-uplets de l'extension. La figure 4 montre comme exemple le résultat du calcul des Concepts de Voisins de Charlotte en partant du graphe de l'arbre généalogique de la famille royale anglaise. On peut y lire que le plus petit concept a pour intention le fait d'être Charlotte et comme extension Charlotte. On retrouve aussi un concept plus large, regroupant tous les enfants de William et Kate (c'est-à-dire Louis et George), ou encore un concept regroupant toutes les femmes (contenant Diana et Kate en plus de Charlotte). Finalement, le concept le plus large a une intension vide et regroupe dans son extension toutes les entités du graphe. On peut voir qu'à chaque concept une distance numérique est associée. Cette distance, aussi appellée *distance extensionnelle*,

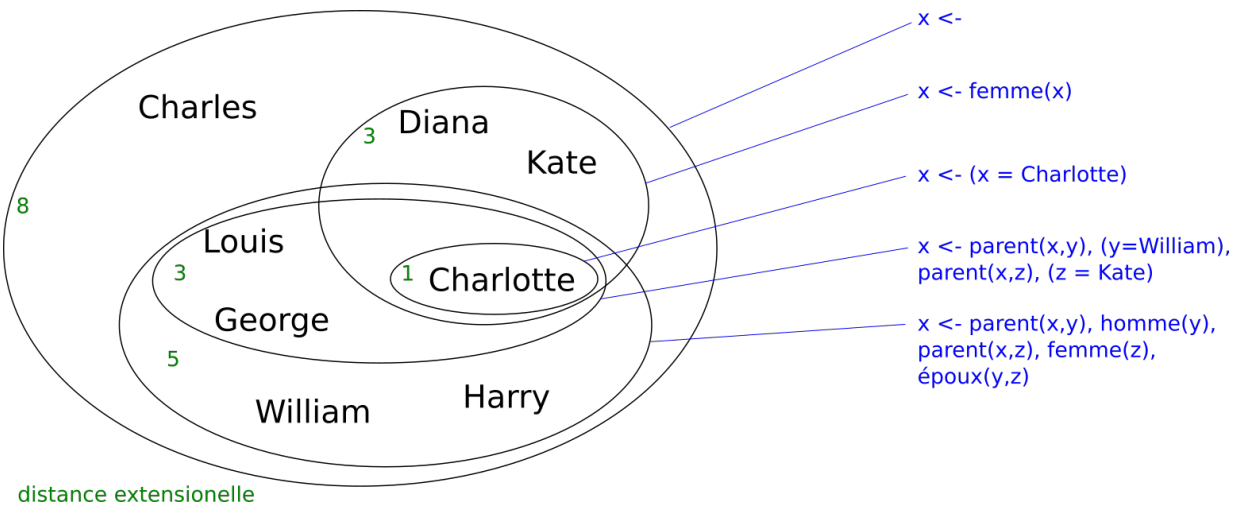

FIGURE 4 – Exemple de Concepts de Voisins

correspond à la taille de l'extension d'un concept, et permet de savoir à quel point un n-uplet est dissimilaire du n-uplet cible. Par exemple, dans l'exemple ci-dessus, William est dans un concept de distance extentionelle 5, il est donc plus dissimilaire de Charlotte que Kate, qui est dans un concept de distance extensionnelle 3.

Un algorithme *anytime* a été proposé pour le calcul de Concepts de Voisins. Cet algorithme part d'un concept généraliste et, par partitionnements successifs, le raffine en concepts plus spécifiques, pour aboutir finalement à l'ensemble des Concepts de Voisins. Ainsi, dans le cas où les calculs sont interrompus avant la fin, l'algorithme est capable de retourner un ensemble de concepts qui est une approximation des Concepts de Voisins.

La méthode proposée est donc la suivante : une fois un texte modélisé selon la modélisation présentée précédemment, pour un exemple de test donné, on calcule les Concepts de Voisins de son couple $(sujet, objet)$ parmi les couples $(sujet, objet)$ de l'ensemble d'entraînement. Puis on applique une méthode de scoring sur les Concepts de Voisins ainsi obtenus, et en regroupant ces scores par relation prédite, on obtient finalement un classement des relations prédites. On peut ainsi prédire que l'exemple de test a pour relation entre son sujet et son objet la relation la mieux classée dans ce classement, ou alors proposer à l'utilisateur les k relations les mieux classées. On constate que cette méthode est bien du *lazy-learning*, différant le calcul des Concepts de Voisins au moment où un exemple est à classer.

De plus, on constate que cette méthode est bien explicable : pour une prédiction donnée, on peut remonter au score de cette prédiction, puis aux Concepts de Voisins à partir desquels a été calculé ce score. L'intension de ces concepts forme alors une explication de pourquoi la relation a été prédite, et l'extension renvoie aux exemples d'entraînement se conformant à cette explication.

Enfin, cette méthode se basant sur la similarité entre exemples, et des exemples ne présentant aucune relation n'ayant aucune raison de se ressembler, elle ne permet pas de trancher si une relation existe ou pas pour un exemple donné, et se contente donc de faire de la *classification de relations* sur des exemples dont on sait qu'ils traduisent une relation.

TABLE 1 – Précision, rappel et F-score pour la détection de relations

Configuration	P	R	F1
luke-base	74.8	79.9	**77.3**
luke-reprod	**76.8**	75.2	76.0
luke-detect	73.1	**80.1**	76.4

4.3 Résultats préliminaires

Ces différents modules ont été évalués en trois temps. Tout d'abord, nous avons évalué le module de détection de relations seul, et avons choisi la configuration la plus adaptée. Ensuite, nous avons évalué conjointement la modélisation et le module basé sur les Concepts de Voisins sur la tâche de classification de relations. Enfin, nous avons superposé ces trois modules afin d'obtenir un système capable de faire de l'extraction de relations à proprement parler.

Ces évaluations se sont faites sur TACRED (Zhang *et al.*, 2017), un benchmark très utilisé en extraction de relations [2]. Ce benchmark est composé de 106 264 exemples (68 124 exemples d'entraînement, 22 631 de validation et 15 509 de test), chaque exemple étant annoté avec une relation parmi 41 possibles ou avec la classe négative *no_relation*. Afin d'être plus proche de la réalité terrain, les concepteurs de TACRED ont fait le choix de laisser 79.5% d'exemples négatifs.

Détection de relations Comme présenté à la section précédente, plusieurs configurations de LUKE pour la détection de relations sont proposées. En plus de *luke-base* et *luke-detect*, nous proposons une troisième variante, *luke-reprod*. Théoriquement équivalente à *luke-base*, elle consiste à reprendre la même configuration, mais à réexécuter l'étape de *fine-tuning*, plutôt que de réutiliser le modèle entièrement entrainé. Cela nous permet de voir si nous rencontrons des problèmes de reproductibilité et, le cas échéant, d'avoir un point de comparaison supplémentaire. Le code utilisé est librement accessible [3].

La table 1 présente les précisions, rappels et F-scores pour ces trois configurations. On peut observer que, bien que théoriquement équivalent, *luke-reprod* ne reproduit pas les résultats de *luke-base*, ce second présentant un F-score supérieur de 1.3 points. LUKE étant implémenté en Python et utilisant CUDA, on peut présumer que cela est dû à un problème de version dans les bibliothèques utilisées ou de CUDA, ou bien que l'entrainement dépend du matériel utilisé. L'entrainement a été réitéré à plusieurs reprises, et a toujours abouti à des résultats similaires. Cependant, on peut constater que *luke-detect* obtient un meilleur F-score que *luke-reprod*. On peut donc supposer que la configuration de *luke-detect* est plus à même de faire de la détection de relations, comme on pouvait s'y attendre, mais que les problèmes de reproductibilité empêchent une comparaison équitable avec *luke-base*.

On peut aussi constater que, bien que *luke-base* ait le meilleur F-score, *luke-reprod* le surpasse en précision et *luke-detect* en rappel. Cependant, privilégier une mesure autre que le F-score implique soit d'avoir un plus grand nombre de faux positifs, soit d'avoir plus d'exemples positifs non détectés, ce qui pose problème lorsqu'on cherche la performance. C'est pourquoi nous adoptons *luke-base* comme module de détection de relations pour la suite.

2. Ce jeu de données n'étant pas libre, nous ne pouvons le partager. Pour plus d'informations, veuillez consulter https://nlp.stanford.edu/projects/tacred.
3. Voir https://gitlab.inria.fr/hayats/luke-redect

TABLE 2 – Taux de bonne classification pour la classification de relations, comparé à la baseline.

Timeout (s)	10	20	30	60	120	300	600	1200
Notre approche	82.0	82.1	82.7	82.9	83.4	**83.6**	**83.6**	**83.6**
Baseline	80.4							

Classification de relations Le code utilisé pour ces expériences est libre et disponible [4]. Il se base sur CONNOR, une implémentation Java de la méthode des Concepts de Voisins [5].

Ces expériences ont été faites sur les exemples positifs de TACRED, c'est-à-dire les exemples annotés avec une relation autre que *no_relation*. De plus, notre système n'ayant pas besoin de jeu de données de validation, le choix a été fait de fusionner les jeux de données de validation et d'entraînement. On obtient au final des jeux de données composés de 18 446 exemples d'entraînement et 3 325 exemples de test.

Comme nous travaillons sur un sous-ensemble de TACRED, il est par conséquent impossible de se comparer directement aux autres approches existantes. Par conséquent, nous nous comparons à une baseline. Cette baseline consiste simplement à prédire, pour un exemple donnée, la relation la plus fréquente dans la base d'apprentissage étant donné le type de son sujet et de son objet. De plus, un score de précision, rappel ou F1 n'ayant aucun sens dans le cadre d'une tâche de classification sans classe négative, nous utilisons le taux de bonne classification.

L'algorithme utilisé étant un algorithme *anytime*, il reste le choix du timeout à faire. Afin de voir l'influence de ce timeout sur les résultats, une gamme de timeouts allant de 10 secondes à 1200 secondes a été testé.

La table 2 présente le taux de bonne classification pour notre approche et la baseline. On peut tout d'abord constater que la baseline atteint un score de 80.4% de bonnes classifications, ce qui est très élevé et montre que le dataset laisse peu de place pour l'amélioration. Ensuite, on peut voir que, quelque soit le timeout, notre approche surpasse la baseline, culminant à un score de 83.6% pour un timeout supérieur à 300 secondes, soit 3.2 points de plus que la baseline. Enfin, on peut observer un phénomène de saturation : on observe un gain de 1.4 points entre 10 secondes et 120 secondes, contre un gain d'uniquement 0.2 entre 120 secondes et 1200 secondes. On peut en déduire que la plupart des concepts sont calculés en moins de 120 secondes, et que le temps supplémentaire sers uniquement à raffiner quelques rares concepts. Cela est confirmé par les chiffres : on observe que, pour un timeout de 10 secondes, seulement 30% des Concepts de Voisins sont totalement calculés, alors que ce chiffre monte à plus de 80% pour un timeout de 120 secondes, et à plus de 99% pour un timeout de plus de 600 secondes. Cela montre qu'en dépit de l'usage d'un algorithme *anytime*, on raisonne en pratique principalement sur les Concepts de Voisins exacts, et non une approximation.

Extraction de relations Nous avons ensuite évalué la combinaison des deux approches précédentes sur la tâche d'extraction de relations. Pour ce faire, chaque exemple de test est d'abord traité par *luke-base*, qui le classifie soit dans la classe *relation*, soit dans la classe *no_relation*. Ensuite, si l'exemple est classifié comme ayant une relation, il est modélisé par le module de modélisation et transmis au module de classification de relations qui se charge de lui attribuer une relation. Les expériences ont cette fois-ci été faites sur l'ensemble du jeu de données TACRED.

4. Voir https://gitlab.inria.fr/hayats/conceptualknn-relex.
5. Voir https://gitlab.inria.fr/hayats/CONNOR.

TABLE 3 – F-score pour plusieurs systèmes d'extraction de relations sur TACRED

Approche	F1 score
LUKE (Yamada *et al.*, 2020)	**72.7**
BERT-LSTM-Base (Shi & Lin, 2019)	67.8
Notre approche	*66.9*
C-GCN (Zhang *et al.*, 2018)	66.4
GCN (Zhang *et al.*, 2018)	64.0

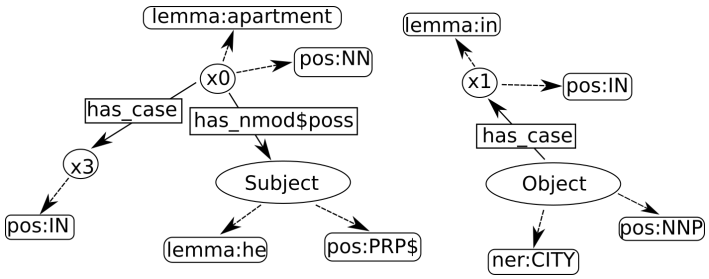

FIGURE 5 – Exemple d'explication

La table 3 présente le F-score micro-moyenné des classes positives de plusieurs systèmes d'extraction de relations, dont le nôtre, sur le dataset TACRED. On peut y lire que notre approche obtient des scores comparables aux approches *deep learning*, se situant entre les approches à base de convolution de graphes (GCN et C-GCN) et les approches utilisant des modèles de langue pré-entrainés (LUKE et BERT-LSTM-Base).

Cependant, le gain principal de notre approche n'est pas quantitatif, mais qualitatif, et repose sur son explicabilité : pour chaque prédiction positive, nous sommes capable de fournir comme explication l'ensemble des Concepts de Voisins ayant été utilisés pour établir cette prédiction. Prenons par exemple la phrase *« Sollecito has said he was at **his** own apartment in **Perugia**, working at his computer. »* Cette phrase est classifiée par *luke-base* comme ayant une relation entre son sujet *his* et son objet *Perugia*, et le calcul des Concepts de Voisins sur la modélisation de cette phrase abouti à la prédiction de la relation *per :city_of_residence*, avec six Concepts de Voisins ayant l'ensemble de leur extension annoté avec cette relation. La figure 5 présente l'intension de l'un de ces concepts. On y voit que ce concept regroupe les exemples ayant comme sujet une entité de lemme *he* qui possède un appartement, et comme objet une ville dans laquelle il y a quelque chose. On peut donc voir que, une fois ce fragment d'explication fourni, la probabilité que la relation soit *per :city_of_residence* est effectivement élevée.

5 Conclusion et perspectives

Dans cet article, j'ai pu présenter le sujet de ma thèse, qui porte sur la conception d'un système centré-utilisateur pour la construction semi-automatisée de graphes de connaissances à partir de textes. Divisé en trois unités, ce système commence par pré-traiter le texte en entrée afin d'en tirer une modélisation sytaxique et sémantique sous forme de graphe. Cette modélisation est alors utilisée par deux unités. La première unité est une unité interactive qui se base sur un module lazy-learning et explicable de classification de relations pour venir faire des suggestions de qualité croissante à

l'utilisateur à travers une interface homme-machine. Enfin, les explications générées par ce module et validées par l'utilisateur viennent nourrir l'unité automatisée, qui les transforme en règles d'inférence et les applique sur la modélisation pour en extraire automatiquement des triplets à ajouter au graphe de connaissances, automatisant ainsi progressivement le processus.

De plus, cet article détaille les éléments de ce *workflow* qui ont déjà été mis en œvre : une modélisation du texte se basant sur la structure syntaxique des phrases et la sémantique des mots la composant, un module d'apprentissage profond pour la détection de relation, et un module basé sur une technologie d'analyse de concepts formels (*Formal Concept Analysis, FCA*) pour la classification de relations. Ces différents modules ont été évalués sur la tâche d'extraction de relations sur le benchmark TACRED, et ont montré des résultats intéressants, aussi bien quantitatifs que qualitatifs.

Enfin, en ce qui concerne les perspectives, des travaux sont en cours sur la transformation d'explications issues du module de classification de relations en règles d'inférence. Une fois cet aspect traité, il restera le développement d'une interface homme-machine remplissant les conditions spécifiées afin d'avoir un système complet, qui pourra alors être évalué via des tests utilisateur.

Remerciements

Ce travail de thèse est fait sous la supervision de Peggy Cellier (INSA, CNRS, IRISA) et Sébastien Ferré (Univ Rennes, CNRS, IRISA).

Références

AYATS H., CELLIER P. & FERRÉ S. (2021). Extracting Relations in Texts with Concepts of Neighbours. In *International Conference on Formal Concept Analysis*.

AYATS H., CELLIER P. & FERRÉ S. (2022). A Two-Step Approach for Explainable Relation Extraction. In T. BOUADI, E. FROMONT & E. HÜLLERMEIER, Éds., *Advances in Intelligent Data Analysis XX*, Lecture Notes in Computer Science, p. 14–25, Cham : Springer International Publishing. DOI : 10.1007/978-3-031-01333-1_2.

DEVLIN J., CHANG M.-W., LEE K. & TOUTANOVA K. (2019). BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT 2019*, p. 4171–4186.

FERRÉ S. (2017). Concepts de plus proches voisins dans des graphes de connaissances. In *Actes IC 2017 28es Journées francophones d'Ingénierie des Connaissances*, p. 163–174.

FERRÉ S. & RIDOUX O. (2002). The Use of Associative Concepts in the Incremental Building of a Logical Context. In *Conceptual Structures : Integration and Interfaces*, Lecture Notes in Computer Science, p. 299–313 : Springer.

GANTER B. & WILLE R. (1996). *Formal Concept Analysis - Mathematic Foundations*. Springer International Publishing.

HITZLER P. (2021). A Review of the Semantic Web Field. *Communications of the ACM*, **64**(2), 76–83.

LYU S. & CHEN H. (2021). Relation Classification with Entity Type Restriction. In *Findings of the Association for Computational Linguistics : ACL-IJCNLP 2021*, p. 390–395, Online : Association for Computational Linguistics. DOI : 10.18653/v1/2021.findings-acl.34.

MAILLOT P., FERRÉ S., CELLIER P., DUCASSÉ M. & PARTOUCHE F. (2017). Nested Forms with Dynamic Suggestions for Quality RDF Authoring. In *International Conference on Database and Expert Systems Applications (DEXA)*, p.1̃6.

MANNING C. D., SURDEANU M., BAUER J., FINKEL J., BETHARD S. J. & MCCLOSKY D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics : System Demonstrations*, p. 55–60.

MARTINEZ-RODRIGUEZ J. L., LOPEZ-AREVALO I. & RIOS-ALVARADO A. B. (2018). OpenIE-based approach for Knowledge Graph construction from text. *Expert Systems with Applications*, **113**, 339–355.

MILLER G. A. (1998). *WordNet : An Electronic Lexical Database*. Cambridge, MA : MIT Press.

MUSEN M. A. (2015). The protégé project : a look back and a look forward. *AI Matters*, **1**(4), 4–12. DOI : 10.1145/2757001.2757003.

NGUYEN T. H. & GRISHMAN R. (2015). Relation Extraction : Perspective from Convolutional Neural Networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, p. 39–48. DOI : 10.3115/v1/W15-1506.

PELLISSIER TANON T., WEIKUM G. & SUCHANEK F. (2020). YAGO 4 : A Reason-able Knowledge Base. In *The Semantic Web*, p. 583–596 : Springer International Publishing.

SHI P. & LIN J. (2019). Simple BERT Models for Relation Extraction and Semantic Role Labeling. *arXiv :1904.05255 [cs]*. arXiv : 1904.05255.

SHNEIDERMAN B. (2020). Human-Centered Artificial Intelligence : Reliable, Safe & Trustworthy. *International Journal of Human–Computer Interaction*, **36**(6), 495–504.

STEINER T., VERBORGH R., TRONCY R., GABARRO J. & DE WALLE R. V. (2012). Adding Realtime Coverage to the Google Knowledge Graph. *ISWC*, p.4̃.

TAI K. S., SOCHER R. & MANNING C. D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *arXiv :1503.00075 [cs]*. arXiv : 1503.00075.

VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER \. & POLOSUKHIN I. (2017). Attention is all you need. In *Advances in neural information processing systems*, p. 5998–6008.

VRANDEČIĆ D. & KRÖTZSCH M. (2014). Wikidata : a free collaborative knowledgebase. *Communications of the ACM*, **57**(10), 78–85.

YAMADA I., ASAI A., SHINDO H., TAKEDA H. & MATSUMOTO Y. (2020). LUKE : Deep Contextualized Entity Representations with Entity-aware Self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* : Association for Computational Linguistics.

ZHANG Y., QI P. & MANNING C. D. (2018). Graph Convolution over Pruned Dependency Trees Improves Relation Extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, p. 2205–2215. DOI : 10.18653/v1/D18-1244.

ZHANG Y., ZHONG V., CHEN D., ANGELI G. & MANNING C. D. (2017). Position-aware Attention and Supervised Data Improve Slot Filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, p. 35–45. DOI : 10.18653/v1/D17-1004.