Synthèse Bibliographique

INSA Lyon Département Informatique

Nejat ARINIK
Xavier RAZANAJATOVO

January 11, 2016

"*Learning to trust is one of life's most difficult tasks.*"
–Isaac Watts

Table des Matières

1 Introduction

De nos jours, le nombre de documents, sites, blogs augmente de plus en plus sur Internet. Grâce à l'utilisation et l'accès généralisé à Internet il est vraiment plus facile d'accéder aux sources digitales qu'aux journaux papiers. De même, nous recevons régulièrement des notifications de journaux par l'intermédiaire d'applications mobiles. Pourtant, la qualité des informations trouvées sur le web décline approximativement aussi vite qu'augmente le nombre des sources digitales.

En effet, il devient de plus en plus courant de publier sur internet, si une fausse information est publiée et qu'on la laisse se propager, l'ensemble des informations est remis en cause et obtenir la réalité devient compliqué. Selon l'article [6], l'une des raisons de la publication de ces fausses informations est l'envie d'être le premier à publier l'information sur internet, particulièrement sur Facebook et Twitter. En général, la majorité des informations est vraie mais une minorité est fausse à cause de la rapidité de la publication. Dans l'exemple de cet article, une personne qui se présente comme un reporter d'information et qui n'a pas beaucoup de followers affirme qu'un crash d'un avion a eu lieu prés d'île Canary à 10h53 sur Twitter et justifie son affirmation en ajoutant la photo du crash. Sa publication a été relayée 600 fois en 10 minutes et s'est propagée malgré que d'autres utilisateurs de Twitter signalaient que c'était un fausse alerte. En effet, l'objet qui apparaissait sur l'image était un remorqueur et non un avion. Cet exemple montre que les rumeurs se propagent effectivement trop rapidement et que cette rapidité a tendance à prendre le dessus sur la qualité journalistique.

Un autre problème qui concerne certaines publications est de cacher certains détails sur l'information. Dans ce cas, l'affirmation est à la fois vraie et fausse selon la condition de l'affirmation. Les détails cachés pourraient être la durée d'une période ou le nom de la ville mentionnée dans l'affirmation et nous aurons besoin d'informations exactes au lieu de celles vagues sans des détails cachés pour vérifier l'affirmation.

Dans tous les cas, les fausses informations ou celles qui cachent des détails sont en général produites intentionnellement afin d'attirer l'attention des lecteurs. Cela se vérifie particulièrement dans le domaine politique, à tel point que nombreux sites spécialisés dans la vérification d'affirmations voient le jour. *Politifact.org* est l'un d'entre eux et se concentre sur la vérification d'affirmations politiques aux États-Unis. En revanche, il n'existe pas beaucoup de journalistes qui cherchent la réalité, vérifient les sources d'information et mènent des enquêtes sur le sujet abordé selon l'article [2].

Le fait que les données digitales augmentent a également un impact négatif pour les journalistes bien qu'ils aient acquéri la possibilité de travailler n'importe où (à la maison, dans le bureau, dans le train etc..) et de publier instantanément ce qu'ils veulent. D'après les publications [2] et [3], les journalistes ont plusieurs problèmes qui les empêchent de travailler efficacement et rapidement.

L'un d'entre eux est l'extraction des informations. En effet, les données appartenant aux journalistes sont généralement non-structurées, ce qui ne leur permet pas tou-

jours d'analyser et de visualiser les interactions entre les différentes sources dans une collection de documents. Les nouvelles ne peuvent donc pas être examinées profondément et les histoires ne peuvent pas être suffisamment élargies.

Se pose aussi le problème de l'exploration des documents. La capacité de classifier les documents de manière intéressante leur permettrait effectivement de faire une bonne synthèse de ce qui existe et de ne pas avoir de documents redondants.

Enfin, l'extraction des données depuis des formes et des comptes-rendus pourrait améliorer l'efficacité du travail des journalistes. Concrètement, ils doivent souvent subir la lecture d'un grand nombre de documents pour vérifier et consolider leurs recherches.

Plusieurs questions qui se posent alors si nous posons le problème du point de vue d'un chercheur informaticien et si nous recourons aux méthodes informatiques. Une plateforme de Fact-Checking peut-elle être un outil d'aide à la vérification des faits? La vérification de faits automatisé est-elle possible? Est-il possible de combiner plusieurs sources digitales? Comment trouver les sources d'informations sur lesquelles l'affirmation repose?

Dans cette synthèse, nous aborderons les trois premières problématiques posées et laisserons la dernière,le sujet de *provenance*, de côté, car elle n'a pas d'impact direct dans ce que nous abordons dans ce sujet. De plus, elle nécessite une recherche complète et détaillée. Sans doute, ce sujet aiderait à l'amélioration de la vérification de faits dans la décennie prochaine.

Notez que nous utiliserons le mot *fait* dans le même sens que *affirmation* tout au long de ce travail; car les deux mots désignent une phrase qui n'est pas toujours vraie.

2 Comment une plateforme de fact Checking peut-elle être aider à la vérification des faits?

Pour une plateforme de fact checking, la vérification des faits dépend d'une assistance humaine. Car, l'utilisateur devrait fournir la (les) base(s) de données sur lesquels la plateforme repose et nettoyer les données erronées en filtrant. En plus, les techniques développés nécessiteraient d'être adaptés à chaque affirmation.

2.1 Approche de Crowd-sourcing

L'approche se base sur 2 points: La première solution est proposée dans l'article [3]. Il s'agit d'un système de *cloud for the crowd* qui combine les ressources informatiques et l'expertise humaine pour aider à un journalisme plus efficace et investigatif. Une partie cloud propose une structure permettant une automatisation des recherches, une simplification des tâches usuelles, ainsi que la collecte et le partage d'informations. De plus, elle permet de planifier et manager intelligemment de grosses tâches en distribuant des sous-tâches à la communauté (la partie crowd) en fonction de leurs préférences et expertise. Mais cette solution dépend des humains et nous risquerions parfois d'obtenir des résultats non satisfaisants ou insuffisants. Dans ce contexte, nous ne pouvons pas parler d'automatisation.

La deuxième solution consiste à se tourner vers les bases de connaissances en ligne construites par des humains. *Wikipedia* est un exemple classique de ce type de dépôt d'informations. Même s'il est construit par des humains, nous faisons, la plupart du temps, confiance aux informations qui y sont présentes. Un exemple serait le projet de *Google Vault* qui est un graphe de connaissance. Quelques articles que nous aborderons dans le sous-chapitre 3.2.3 utilisent ce graphe pour faire de la prédiction de liens.

2.2 Comment trouver des questions intéressantes à partir d'une affirmation?

La question qui est posée par les articles [3] et [18] est de *Trouver des questions à partir des réponses des requêtes* afin d'explorer les bords d'une affirmation. Pour cela, l'article [3] propose 3 approches: rendre les histoires *live* (mises à jour continuellement), rendre les histoires *multiples* (rendre les paramètres génériques) et vérifier les informations (*Fact-checking*).

La première approche est théoriquement bonne, mais n'est en réalité pas facile à mettre en œuvre et n'est pas applicable à tous les types d'affirmation. Elle serait par exemple utile dans le cas d'une étude du taux de pollution de l'eau. Il serait possible d'exécuter une requête de façon journalière plutôt qu'annuelle dans une base de données mise à jour régulièrement pour contrôler le niveau de saleté. En cas d'augmentation du taux au dessus d'un seuil défini, le système pourrait nous

prévenir en temps réel.

Pour les autres types d'affirmations, les auteurs s'intéressent aux approches 2 et 3 qui constituent en fait la base des articles [18] et [19]. Dans l'article [18], on modélise une affirmation comme une requête paramétrée qui permettrait de créer d'autres histoires en rendant les paramètres génériques. Selon l'article, est définie une requête initiale: $< q, p_0, r_0 >$ où q est la requête, p_0 est un ensemble de paramètre basé sur l'affirmation et r_0 est le résultat original. On perturbe les paramètres et on obtient: $< q, p, r >$ où p et r sont respectivement l'ensemble des paramètres et résultat final. C'est-à-dire qu'en perturbant les paramètres, on génère des histoires différentes. Ce travail perturbe les affirmations en faisant du reverse-engineering et en trouvant des contre-arguments. Le but du reverse-engineering est de clarifier les affirmations vagues et de trouver d'éventuels détails cachés. Trouver des contre-arguments a pour objectif d'affaiblir l'affirmation originale afin de vérifier s'il y a des combinaisons de paramètre intéressantes par rapport au contexte de l'affirmation. Finalement, si r et r_0 sont considérablement différents (un écart), l'affirmation peut être considérée comme fausse. Dans l'article, nous pouvons visualiser les résultats de différentes combinaisons de paramètres sur une charte de chaleur, appelée Query Response Surface. Nous voyons un exemple dans la figure 1 où les paramètres associées aux cases rouges ont un grand écart négatif par rapport à la paramètre originale. Pourtant, celles associées aux cases vertes ont un écart positif, c'est-à-dire elles consolident l'affirmation.Par exemple, si l'affirmation parle d'une augmentation d'adoption de 65%, une case verte indique une augmentation de plus de 65% comme 70%. Ainsi, on évalue l'affirmation originale en vérifiant si elle est correcte et dans quelle condition elle est correcte.

Mais dans cet article, les auteurs s'intéressent surtout aux affirmations correctes mais qui cachent des détails. C'est leur définition d'*intéressant* dans ce contexte. Nous voyons que le modèle conçu est utilisable pour tous les domaines, mais qu'il convient particulièrement aux affirmations politiques.

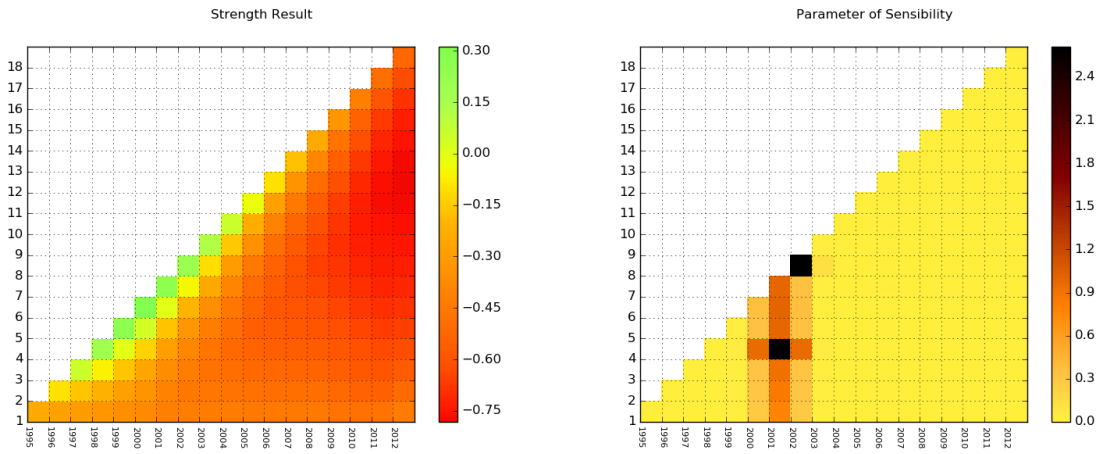

Figure 1: Visualisation des résultats de différentes combinaisons de paramètres d'une affirmation

D'un autre côté, l'article [19] modélise ces 2 approches spécifiquement dans les do-

maines où les objets jouent un rôle important, et notamment le domaine du sport (basket-ball) ou dans des domaines similaires comme *Computer Science Bibliography* et *Wikipedia Edit History*. L'objectif est de profiter de la puissance de visualisation des données en 2 dimensions afin de distinguer les performances modestes et extraordinaires des objets. Dans ce contexte, les performances modestes et extraordinaires correspondent respectivement aux *outliers* et aux *clusters* qui sont deux indices visuels essentiels. La logique est assez simple: il est rare qu'un joueur de basket-ball marque au moins 50 points, nous considérons donc cette performance comme une anomalie. Les autres performances se classifient selon leurs similarités. Les affirmations qui annoncent des hautes performances visent en général à décrire un fait de manière exagérée pour en souligner le caractère exceptionnel afin d'attirer l'attention des lecteurs et téléspectateurs. En ce qui concerne les 2 approches, le système prend en entrée les performances de chaque joueur dans une période définie en se basant sur 2 dimensions, par exemple le couple (points, rebounds) en basket-ball, et sort les résultats représentés par la combinaison sous forme de zones colorées en fonction de la densité de points présents et de points isolés. Puis, on évalue l'affirmation selon la visualisation et vérifie la cohérence et la justesse de l'affirmation.

2.3 Comment les approches de Fouille de Données peuvent-elles contribuer au Fact Checking?

Une autre solution pourrait être de se tourner vers les méthodes de *fouille de données*. Par exemple, la représentation sous forme de graphe permettrait de vérifier la véracité d'une affirmation et de trouver des détails cachés en explorant les noeuds. Il y a plusieurs applications à la fouille de données comme la recherche des itemset fréquents, l'extraction des données, la détection d'anomalie etc...

En ce qui concerne le fact-checking, l'extraction de données intéressantes pourrait aider à vérifier des affirmations à partir de grandes bases de données. Cette idée construit le sujet de *découverte des connaissances dans des bases de données* (*KDD*). C'est la raison pour laquelle nous nous intéressons au rapport de recherche [13]. Il introduit des méthodes symboliques de KDD telles que la *classification à base de treillis*, la *recherche des itemsets fréquents* et la *recherche de règles d'association*.

La *classification à base de treillis* se base sur l'analyse de matrices booléennes et permet d'extraire l'ensemble de concepts organisés dans une hiérarchie. Par exemple, un ensemble d'objets et un ensemble d'attributs seraient les axes de la matrice booléene et les valeurs binaires correspondantes entre les 2 axes pourraient être les valeurs dans la matrice booléenne. Cela peut être vu comme un treillis de Gaulois d'une relation binaire. Un treillis est un ensemble partiellement ordonné dans lequel chaque couple d'éléments admet une borne supérieure et une borne inférieure. Un treillis peut être construit par la correspondance de Gaulois dans une analyse formelle de concepts. L'analyse de concepts formels (ou juste concepts) s'attache à étudier les concepts lorsque le contexte et les concepts sont complètement et précisément définis. Un concept peut être défini par son intention et son extension: L'extension est l'ensemble des objets qui appartiennent au concept tandis que l'intention est

l'ensemble des attributs partagés par ces objets.

Un treillis de Gaulois comprend tous les motifs fermés (appelé concept formel aussi). Autrement dit, les motifs fermés (*closed itemset*) sont l'ensemble maximal d'objets qui partagent le même ensemble d'attributs. Par exemple, dans la figure 2, le motif *1, acd* indique que l'objet 1 se trouve dans les attributs a, c et d et il n'y a aucun objet ayant les mêmes attributs car il apparaît dans le treillis.

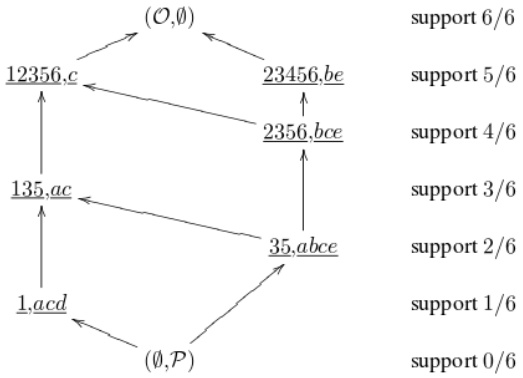

Figure 2: Un treillis de Gaulois
[11]

La fouille des motifs fermés à partir des matrices booléennes est étudiée en détail par l'article [16]. Par exemple, dans une base d'éléments donnée, si $P = B, C, D$ est un motif fermé associé à l'ensemble d'objets $O = 1, 2, 3$, cela veut dire qu'on ne peut ajouter aucun attribut à P qui serait correct pour tous les objects dans O.

Même si le treillis comprend tous les motifs fermés, le nombre de ces motifs pourrait être considérablement élevé. Néanmoins, on peut trouver des motifs fréquents à partir des motifs fermés situés dans le treillis. Comme le treillis est organisé par support croissant, il suffit de parcourir les sommets du treillis jusqu'à ce que le nombre d'objets soit inférieur au minimum support (seuil prédéfini).

Malgré plusieurs travaux sur la fouille de relation binaire comme la matrice booléenne, la plupart des bases de données correspond à une n-ary relation où $n > 2$. Par exemple, à coté d'*objets* et d'*attributs*, il peut y avoir des *dates* et des *places* dans une 4-ary relation. Cette modélisation permet de ne pas perdre d'informations lors des projections et agrégations appliquées. L'un des articles qui propose cette modélisation est [14]. Dans cet article, les auteurs définissent de nouvelles règles d'association pour la n-ary relation. Selon eux, la sémantique de leurs règles est moins contraignante par rapport à celle des travaux précédents. De plus, ils implémentent un algorithme qui trouve toutes les règles intéressantes de manière exhaustive et extensible en prenant en compte toutes les dimensions et non une seule. Ils valident leur travail avec les données de $Velov$ en les modélisant comme un graphe dynamique orienté de dimension 4 (4-ary).

Même s'il n'existe pas d'application de tensor booléen sur le fact-checking, on peut très bien imaginer une plateforme qui l'utilise. Si on considère un tensor comme un matrice 3D comme dans la figure 3, chaque matrice 2D correspondra à une

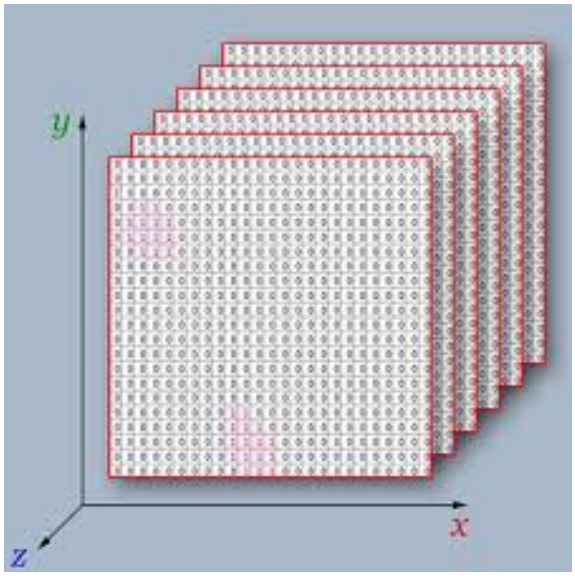

Figure 3: Matrice 3D pour concevoir un tensor booléen

base de données. Par exemple, le premier appartient aux données de criminalité, le deuxième appartient aux données de chômage et on peut trouver qu'il y a une corrélation positive entre le nombre de chômage et le nombre de crime dans un zone de quartier. Grâce à l'aide d'un tensor booléen, nous pouvons combiner plusieurs bases de données et nous aurions plusieurs points de vue pour évaluer une affirmation.

Le treillis et l'itemset sont des concepts similaires. En effet, la *recherche des itemsets fréquents* correspond à une recherche de *Breadth-First-Search* dans le treillis de Gaulois. La recherche des motifs fréquents fait l'hypothèse d'une base de données décrivant un ensemble d'objets $O = \{o_1, ..., o_m\}$ par un ensemble d'attributs $A = \{a_1, ..., a_n\}$. Un objet o est décrit ou non par un attribut a de sorte que la base de données est assimilable à une relation binaire. Pour un ensemble de données O et l'ensemble d'attributs A, un item est un attribut d'un objet et un itemset (en d'autres termes, un motif) est un ensemble d'items. Le support d'un itemset est la proportion des objets contenant l'itemset en respectant toute la population des objets. Un itemset est dit fréquent s'il apparaît au moins N fois où N est un seuil prédéfini. Les algorithmes de recherche des motifs fréquents doivent parcourir la totalité de la base de données chaque fois qu'ils doivent déterminer la fréquence d'un ou de plusieurs motifs.

L'extraction des ensembles fréquents de lots de données est souvent présentée comme une collection de règles, appelées règles d'association. La forme d'une règle d'association est $I \rightarrow j$; où I et j sont 2 ensembles d'éléments (itemset). L'implication de cette règle d'association est que si tous les éléments dans I apparaissent dans certains panier, alors j est *probable* à apparaître dans ce panier. Une exemple possible serait *ceux qui ont acheté du pain et du lait ont aussi acheté du beurre*. La génération des règles d'association valides se fait de manière similaire à la recherche d'un itemset fréquent.

L'usage du modèle des règles d'association en fouille de données est limité par la quantité prohibitive de règles qu'il fournit et requiert la mise en place d'un post-traitement efficace afin de cibler les règles les plus utiles. L'article [12] propose une nouvelle approche de fouille de règles d'association qui intègre explicitement les connaissances de l'utilisateur. Les connaissances de l'utilisateur sont modélisées à l'aide d'une ontologie associée aux données traitées et de schémas de règles, qui apporte des connaissances supplémentaires sur les données. Les schémas de règles permettent de définir une forme caractérisant les règles intéressantes à sélectionner parmi l'ensemble des règles calculables.

3 La vérification des faits automatisé est-elle possible?

Nous avons insisté sur la vérification des faits si l'utilisateur joue un rôle important dans le processus de la vérification. Pourtant, il existe d'autre travaux qui se focalisent sur l'extraction des faits automatiques.

3.1 Détection des faits automatiques

Tout d'abord, nous allons parler des articles qui visent à trouver des affirmations intéressantes automatiquement (sans que l'utilisateur ne fournisse l'affirmation) à partir d'un texte, c'est-à-dire des documents non-structurés. Ensuite, nous traiterons le cas d'autres articles qui visent à trouver des affirmations à partir d'une source auditive, notamment un discours en temps réel.

3.1.1 A partir d'un texte

L'article [9] nous expose une plateforme qui détecte des affirmations intéressantes dans un document et qui donne un score de justesse et de *check-worthy*. Pour donner ce score, l'outil utilise sa base de données composée d'affirmations précedemment analysées par le système, tirées des débat présidentiels aux Etas-Unis ces 30 dernières années. A partir de la base de données, la plateforme fait une apprentissage supervisé en catégorisant les phrases en 3 parties:

- *Non-Factual Sentence (NFS)*: Les phrases subjectives (opinions, croyance, déclarations) et des questions.
- *Unimportant Factual Sentence (UFS)*: Les phrases qui sont de type affirmatives mais qui ne valent pas la peine d'être vérifiées. Par exemple: *Demain est le jour d'élection.*
- *Check-worthy Factual Sentence (CFS)*: Les affirmations qui valent le coup d'être vérifiées. Par exemple: *Plus d'un million d'américains sont HIV-positive.*

Par ailleurs, le système est capable de fusionner quelques phrases pour les considérer comme une seule affirmation. Cela s'avère très utile, car dans la pratique, il n'est pas rare de rencontrer de longues affirmations ou des affirmations composées de plusieurs phrases. De plus, la plateforme permet de visualiser ces résultats.

Comme la plateforme présentée dans l'article [9], l'un des graphes de connaissance *Google Vault* est également capable d'analyser un texte pour détecter des faits et les vérifier. D'abord, il analyse le document phrase par phrase. Puis, il extrait les RDF et les ajoute au graphe.

De la même manière, l'article de démonstration [5] propose aussi une plateforme qui détecte des affirmations de 3 types automatiquement à partir des données *structurés*. Les types sont *Situational facts*, *One-of-the-few claims* et *Prominent streak factlets*. En fait, les auteurs des articles [19] et [18] développent leur travail. De plus, la plateforme permet aussi la visualisation des résultats.

D'autre part, l'article [8] propose un outil d'aide au processus d'analyse d'information, appelé *fact checking and analysis*. En plus de l'extraction automatique de données, cette plateforme permet à l'utilisateur d'annoter les documents semi-structurés avec des informations sémantiques pour améliorer les résultats de la plateforme. Cela requiert une modèle de données permettant la représentation de telles données et un langage de requête pour en extraire les informations. Pour cela, FactMinder utilise le modèle de données XR ainsi que le langage de requête qui lui est associé, XRQ, dont la description et le fonctionnement sont détaillés dans le document.

Le processus du système, *FactMinder*, est la suivante: Le client de *FactMinder* est constitué de 3 composants, un information exctractor, un rich browser et un dashboard, chacun jouant un rôle différent dans le processus de fact checking. Lorsqu'un document est ouvert dans le client, le module d'information extraction récupère automatiquement les sujets, les entités, et les relations qu'ils contiennent. Les documents sont ensuite ouverts dans le rich browser où l'utilisateur pour manuellement ajouter ou modifier des annotations. Le dashboard est composé de vues de la base de données XR avec les informations annexes. Pour effectuer ses investigations, l'utilisateur bascule entre le rich browser et le dashboard en ajoutant des détails au document via de nouvelles annotations.

3.1.2 Défis en temps réel

De nos jours, on peut considérer qu'un système ne donnant pas de résultats en un court laps de temps est inefficace et non performant. De plus, créer un système en temps réel est un défi important pour le journalisme informatique. C'est la raison pour laquelle les articles [18] et [19] s'intéressent à l'efficacité de leurs algorithmes et proposent à la fois un algorithme de base et un ou plusieurs algorithmes performants. Le fait d'implémenter un algorithme de base permet de comparer ses performances avec celles de l'algorithme optimisé.

3.2 Vérification des faits automatiques

Une fois que nous avons détecté les faits, nous pouvons passer à l'étape de vérification. Nous présentons cet étape en 2 partie: la catégorisation des faits et l'approche sous forme de graphes de connaissances. Pour cela, il faut distinguer la différence entre une bases de données et une bases de connaissances. Lorsqu'on interroge une base de données, on n'obtient que des informations qui y sont déjà de façon explicite. Lorsqu'on interroge une base de connaissance, on peut obtenir des informations qui n'y sont pas explicitement. Par exemple, supposons une règle comme la suiv-

ante: *père(x,y)* ⇒ *parent(x,y)*. Là, *père(Marcèl, Maurice)* est un exemple de *fait*. Et sans la règle, nous ne pourrons pas conclure *parent(Marcèl, Maurice)* car nous l'obtiendrons à partir de ce fait.

3.2.1 Catégorisation des affirmations

Afin d'avoir un système automatisé, il faudrait classifier les types d'affirmations et les généraliser avec une formule mathématique. Dans l'article [18], les auteurs s'intéressent aux 2 types d'affirmations: *Window Aggregate Comparison Claims* et *Time Series Similarity Claims*. Dans le cas de *comparaison de fenêtre agrégée*, nous nous intéressons aux actions d'une seule personne, ou plus généralement d'une entité, entre deux périodes de même longueur sur un sujet quelconque. Par exemple, si un président compare son mandat et celui du président précédent, on utiliserait ce type de catégorisation. Pourtant, dans le cas de *similarité de séries temporelles* nous nous intéressons aux actions de deux entités, entité source et cible, dans la même période. Par exemple, si un député compare son nombre de vote commun avec un autre député dans une période fixe, on utiliserait ce type de catégorisation.

D'autre part, les auteurs de l'article [19], catégorisent 3 types de requêtes:

- Une *Projection Query* permet de trouver des statistiques à propos d'un objet pour chaque action. Par exemple en basket-ball, si nous nous intéressons aux performances d'un joueur en prenant en compte lee nombre de points et le nombre d'assists (passe décisive) pour chaque match, l'un des résultats de cette requête serait (31, 31) correspondant au match d'un basketteur ayant marqué 31 points et effectué 31 assists.
- Une *Count Query* permet de trouver le nombre d'actions associées à des statistiques dans la totalité des données. Dans le cas du basket-ball, la paire (50, 38) peut correspondre à 38 matchs dans une saison avec au moins 50 points marqués.
- Une *Streak Query* permet de trouver une série d'actions associées à des statistiques dans la totalité des données. Dans le cas du basket-ball, la paire (35, 9) peut correspondre à 9 matchs consécutifs dans une saison avec au moins 35 points marqués.

En plus des types de requêtes des 2 articles précedemment mentionnés, l'article [5] distingue les *situational facts*, ou faits de situation. Un fait de situation se rapporte à un objet qui se révèle plus impressionnant que les autres dans un contexte précis (les photos postées sur Facebook), lorsqu'ils sont comparés par plusieurs mesures (le nombre de like, de commentaires etc.). Par exemple: *La photo la plus virale diffusée sur les réseaux sociaux a généré 3,500,000 de likes, 170,000 commentaires et 460,000 partages depuis mercredi après-midi.*

Tous les articles abordés dans cette partie visent à créer un système facile à utiliser, notamment pour les journalistes, et où les types de requêtes mentionnés sont considérés comme une fonction entrante du système. Les articles [18] et [19] développent

un système qui prend en entrée le type d'affirmation et l'affirmation en elle-même, puis produit un résultat avec éventuellement sa qualité dans une base de données précise. Au fur et à mesure de l'utilisation du système, une véritable librairie de données et de requêtes sera créée et pourra être utilisée pour construire une *reporter's black box*, outil permettant de lancer des requêtes standards, qu'il aura évaluées comme les plus pertinentes sur telles données.

Cependant, les solutions proposés par [18] et [19] ne sont applicables que dans une base de données et cette base de données doit être fournie. Par ailleurs, leurs systèmes ont besoin de connaître et modéliser tous les types d'affirmations alors que ce n'est pas le cas. Néanmoins, ils modélisent les types d'affirmations fréquemment utilisés sur internet. Donc, leur système a un potentiel d'être automatique.

3.2.2 Défis en temps réel de la catégorisation des affirmations

Les auteurs de l'article [18] propose à la fois des méthodes de brute-force et leurs versions optimisés afin de confronter au défi de real-time; car l'objectif idéal est de proposer un système automatisé et rapide.

Pour cela, l'article propose 3 méta-algorithmes dans son système dont 2 sont des versions optimisées de l'algorithme de base. L'algorithme de base essaie normalement tous les paramètres possibles de manière exhaustive afin de trouver des contre-arguments ou faire du reverse engineering, mais un problème d'espace est rencontré. Deux méthodes sont proposées pour palier à ce problème: L'*énumération ordonnée de paramètres* génère des paramètres avec un SP élevé. Cela permet de se concentrer sur une partie de paramètres si l'espace est suffisamment large. Soit on énumère les paramètres pour chaque dimension (temps, distance de période), soit pour chaque critère (idéologie des entités). L'algorithme "Diviser et conquérir sur l'espaces de paramètre" n'a pas besoin de calculer tous les paramètres. D'abord l'algorithme divise l'espace P en zones, puis trouve la meilleure configuration de paramètres de chaque zone. Cet algorithme est plus efficace et utile par rapport aux précédents. On obtient à la fin une liste de paramètres utilisables en tant que contre-arguments ou pour faire du reverse engineering.

D'un autre côté, l'article [19] propose un algorithme optimisé par rapport à son algorithme de base. Le *Baseline algorithm* effectue une évaluation exacte de la requête d'exploration basée sur la totalité des données. Il prend en entrée le type de fonction de requête avec ses attributs spécifiques et calcule dans toutes les données les points rares (*sparse points - outliers*) et en déduit les points restant (*clusters*) selon les résultats de la fonction. Le *Sampling-based algorithm* est sa version optimisée qui produit une solution approximative plus efficace. Cet algorithme procède à une sélection d'objets, mais cette sélection ne doit pas être totalement arbitraire, certains points d'intérêt (les sparse points notamment) ne doivent pas être exclus du traitement. Le challenge est ici de déterminer ces points en fonction de la requête à traiter. Ces points constituent une partie de l'échantillon de données et sont complétés par des données aléatoire récupérées dans le lot de données restant. La méthode de sélection d'objets pour l'exécution de cet algorithme le rend plus performant que

celui traitant toutes les données.

Parallèlement à cela, la plateforme de l'article [9] est utilisable pour traiter un discours en temps-réel. Pour cela, elle utilise un outil existant pour convertir le discours en phrases. Leur qualité dépend donc de cet outil. De plus, les auteurs ne parlent pas de la performance de leur plateforme. Nous ne sommes donc pas en mesure de vérifier si leur plateforme correspond bien au défi du temps-réel.

3.2.3 Approche de Knowledge Graph

Nous avons jusqu'à présent abordé des méthodes qui nécessitent d'être accompagné par un expert qui guidera toutes les phases de l'exécution des algorithmes. De plus, il faudrait fournir en entrée une ou des bases de données. Néanmoins, elles sont assez utiles et permettent d'analyser des affirmations de manière efficace et rapide, particulièrement pour un journaliste.

D'autre part, il existe des articles qui profitent des dépôts ou encyclopédies d'informations pour vérifier un fait automatiquement. *Wikipedia*[1] est un exemple classique de ce type de dépôt. L'idée de l'article [1] est de modéliser les données semi-structurées qu'un dépot de connaissance possède comme un graphe de connaissance, *Knowledge Graphs* en anglais. Dans cette modélisation, un *fait* est défini par un triplet de (*sujet*, *prédicat*, *objet*) comme ("United Flight U323", "has departure time", "16:10") dans la figure 4. Et donc l'ensemble de faits construit un graphe de connaissance. Les *objets* et les *sujets* des triplets sont représentés par des nœuds et les *prédicats* sont représentés par les liens.

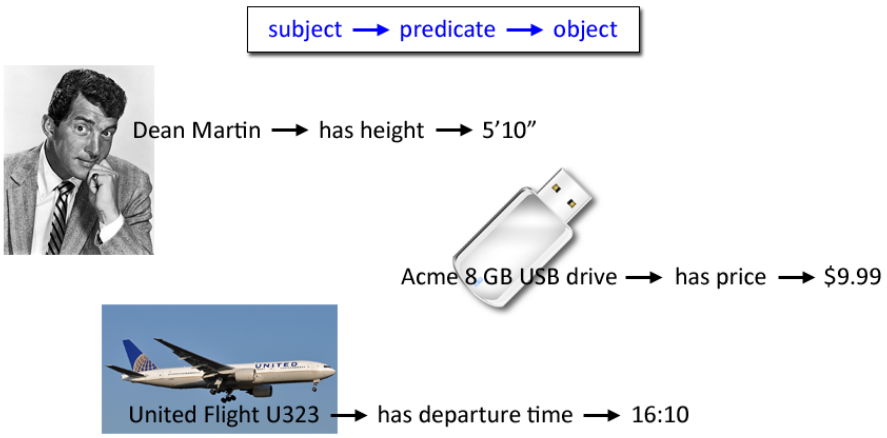

Figure 4: Exemples de RDF

Dans l'article, un *fait* modélisé sous forme d'un triplet est correct s'il existe un court chemin entre le sujet et l'objet avec k nœuds intermédiaires où k est donné. Dans cette approche, les auteurs définissent que la *proximité semantique* et la justesse d'un fait dépendent de la maximisation de la proximité sémantique. De plus, ils prennent en compte le fait que chaque lien a son propre *fact support* (support de

[1] https://www.wikipedia.org/

fait), c'est-à-dire que même s'il y a 2 triplets ayant les mêmes sujets et les mêmes objets et si leurs prédicats sont différents, les 2 faits ont de différents *fact support*. Par exemple, (*John, playedIn, Star Trek*) et (*John, starredIn, Star Trek*) ont de différents *fact support*.

Cette approche est loin d'être performante car il y a des milliards de possibilités dans la construction d'un fait et un graphe de connaissance ne peut pas contenir toutes les informations. Pour pallier à cela, l'article [7] propose des méthodes pour ajouter de nouveaux liens et supprimer d'éventuelles erreurs dans le graphe de connaissance.

L'approche qui y est proposée est d'utiliser la fouille des règles de Horn en introduisant un modèle formel pour la *fouille de règles* sous l'hypothèse *Open World*. Selon cette hypothèse, une déclaration de fait (statement of a fact) qui ne se retrouve pas dans le graphe n'est pas nécessairement incorrect. Il est juste inconnu. Alors que si le prédicat n'existe pas, le fait est incorrect sous l'hypothèse de *Closed World*. Par exemple, s'il n'y a pas de prédicat *mariedTo* pour une personne donnée dans un graphe, cette personne est célibataire sous l'hypothèse de Closed World. Mais nous ne pouvons rien dire sous l'hypothèse Open World car elle peut être célibataire ou bien mariée.

Les méthodes proposées que nous avons traitées se basent sur les propriétés de graphe. L'article [15] propose plusieurs techniques statistiques de *Statistical Relational Learning*, SRL, à appliquer sur une base de connaissance (knowledge graph) à grande échelle. Le but principal du SRL est la prédiction des liens manquants, prédiction des propriétés de nœud et clustering de nœuds à base de *connectivity patterns*.

D'après l'article, il y a des règles déterministes telles que les contraintes et la transitivité. Pourtant, il y a dans les bases de connaissances des motifs statistiques qui ne sont pas toujours vrais mais qui sont utiles pour la prédiction de liens. Un des exemples est l'*homophilie*, le fait que des entités soient similaires à d'autres entités d'une manière quelconque à par l'intermédiaire de quelques caractéristiques personnelle. Pour les données multi-relationnelles (multiples types de labels sur les liens), l'homophilie est appellé aussi *autocorrelation*. Par exemple, *Les acteurs nés aux Etas-Unis sont plutôt célèbres dans les films américains*. Un deuxième exemple est la similarité de structure. Les liens manquants pourraient être complétés avec cette méthode. Par exemple, s'il existe les triplets (Alex, playedIn, StarWars) et (Alex, starredIn, StarWars), et s'il existe le triplet (Guiness, starredIn, Starwars), on peut prédire le triplet (Guiness, playedIn, Starwars). Par contre, ce n'est pas toujours vrai. Dans le contraire, c'est-à-dire que s'il existe les triplets (Alex, playedIn, StarWars) et (Alex, starredIn, StarWars), et s'il existe le triplet (Guiness, playedIn, Starwars), on ne peut pas prédire que le triplet (Guiness, starredIn, Starwars) existera.

Selon l'article, l'absence (ou présence) de certains triplet est corrélés avec d'autre triplets présents (ou respectivement absents). Les auteurs modélisent donc les exemples de corrélation mentionnés en 3 parties: *Modèle de Caractéristique Latente*, *Modèle de Caractéristique de Graphe* et *Combinaison des 2 Modèles Précédents*. Une exemple du modèle de caractéristique *latente* serait le suivant: *Alec Guiness*

remporte le prix d'académie britannique Une explication serait *Alec Guiness est un bon acteur. C'est pourquoi il remporte le prix.* Comme on peut le voir, la première phrase ne suffit pas à conclure qu'Alec Guiness est un bon acteur. Alors ici, *être un bon acteur* est une caractéristique latente. Le modèle de caractéristique à base de graphe est similaire aux travaux précédents. De plus, les auteurs combinent les 2 modèles pour avoir un meilleur résultat.

Nous pouvons dire qu'un graphe de connaissance n'est pas complet mais la plupart des liens y sont corrects. Avec cette hypothèse, les 2 derniers articles mentionnés (dans leurs parties de "link prediction") concluent qu'un fait est correct s'il se trouve dans le graphe ou il est accessible par k nœuds intermédiaires. Cependant l'article [17] stipule que la prédiction de lien ne suffit pas d'elle-même pour la vérification de faits. En effet, les méthodes proposées dans cet article ne vérifient pas seulement la justesse d'un fait mais expliquent aussi le sens de quelques relations entre noeuds. Dans leur approche, une minorité des liens/chemins est utile pour la vérification même s'il y a une masse de données dans le graphe. Ils développent donc un algorithme de *discriminative path mining* pour récupérer des *discriminative meta paths* et *predicate paths* qui seront utiles à la prédiction de liens. Un *meta path* est défini par une séquence de nœuds et de liens labellisés et orientés (directed) comme l'expression 1 où p représente un prédicat et o représente un objet:

$$\left\{ O_1 \xrightarrow{p_1} O_2 \xrightarrow{p_2} O_3 \right\} \tag{1}$$

Un *predicate paths* est une séquence de prédicats pour obtenir le fait (du sujet à l'objet). Dans l'expression 1, un chemin de prédicat est $\left\{ p_1 \rightarrow p_2 \right\}$. Un *discriminative path* est un meta path à longeur de k noeuds où k est le longueur maximum.

Le système se déroule en 3 phases: *Extraction*, *Selection* et *Validation*. Dans la phase d'extraction, des metapaths qui relient alternativement les objets et les sujets (dans le triplet) sont collectés. Puis, on trouve les discriminative meta paths les plus importants à partir des meta paths extraits dans la phase de sélection. A la fin, une comparaison des (sujet, prédicat, objet) est fait par l'intermédiaire du modèle de régression construit par des discriminative paths pour comprendre comment ils se sont liés entre eux et par quels chemins.

Ce sont des *discriminative meta paths* qui sont utilisés pour la vérification de faits mais cela ne veut pas dire que tous les *predicate paths* décrivent les caractéristiques/attributs les plus importants et intuitifs. Alors, au lieu de s'intéresser à tous les predicate paths, l'extraction d'un sous-ensemble de predicate paths est réalisée, nommé *discriminative predicate paths*. Ainsi, nous trouvons les predicate paths les plus *importants* et supprimons tous ce qui n'est pas nécessaire. En fait, les chemins de prédicats les plus importants sont des chemins qui sont rarement utilisés par d'autres faits. Avec cette définition d'*intéressant* l'algorithme de cet article est capable d'expliquer le sens de quelques relations entre noeuds.

3.3 Qualité de la vérification de faits

3.3.1 Mesures de qualité d'une affirmation

L'article [18] traite de la qualité d'une affirmation. Selon l'article, des différents mesures de qualité permettraient de voir quelqeus perspectives sur l'évaluation d'une affirmation. Les auteurs y ont défini 3 mesures de qualité:

- *Solidité (Robustness)*: Les valeurs possibles sont dans une intervalle $[0, 1]$. Une valeur de 1 signifie que toutes les perturbations résultent des affirmations fortes. Une valeur proche de 0 que l'affirmation originale peut facilement être affaiblie
- *Unicité (Uniqueness)*: Une mauvaise valeur d'unicité signifie qu'il est facile de trouver des affirmations perturbées aussi fortes que l'affirmation originale
- *Justesse (Fairness)*: Les valeurs positives désignent que l'affirmation est exagérée. Les valeurs négatives désignent que l'affirmation est amoindrie, c'est-à-dire que l'importance de l'affirmation est minimisée. Et 0 désigne l'impartialité

Les trois mesures de qualité sont définies en fonction de leur *résultat de force relative* (*SR* à partir de maintenant) et de leur *paramètre de sensibilité* (*SR* à partir de maintenant). Le SR sert à mesurer la force d'une affirmation en comparant le résultat initial et le résultat après que l'on ait appliqué des perturbations. Le SP sert à mesurer à quel point l'affirmation est naturelle et pertinente (*naturalness* et *relevancy* respectivement). Le fait d'être pertinent dépend du contexte de l'affirmation et le fait d'être naturel concerne les domaines de paramètre du contenu. Par exemple, si on parle de la période de mandat d'un maire qui est 4 ans, la durée de 4 ans reflète la qualité du naturel de l'affirmation. Mais il se peut que l'affirmation soit focalisée sur une période de 6 ans. Donc ici, la durée de 6 ans reflète la qualité de sa pertinence. Même si les fonctions SP et SR sont bien définies et donnent de bonnes résultats, une autre approche pourrait être de recourir à l'apprentissage supervisée pour aider à spécifier les fonctions SP et SR. L'idée est de s'adresser aux crowdsourcing en se reposant sur le feedback des utilisateurs. Par ailleurs, les mesures de qualité ne donnent pas toujours une explication compréhensible. Le problème est qu'elles se basent sur les fonctions SP et SR. Il serait mieux de développer ces mesures avec d'autres notions.

3.3.2 Qualité d'une source pour la vérification de faits

Les bases de connaissances construites automatiquement sont bonnes si les sources à partir desquelles les informations (triplets) sont extrait sont bonnes. L'article [10] fait l'état de l'art sur la fusion de sources multiples pour mesurer la justesse des sources. Parce que la fusion des données pourrait permettre de résoudre les conflits et trouver la valeur correcte sur le Web.

La plupart des approches de fusion de données suivent l'approche suivante: on récupère d'abord les données pour chaque élément comme la température d'une ville ou l'heure de départ d'un vol depuis plusieurs sources. On affecte ensuite les valeurs les plus reçues aux éléments et on considère la valeur affectée comme une valeur correcte. Par exemple, si on reçoit 15h50 depuis la majorité des sources et 16h20 pour la minorité pour l'heure de départ d'un tel vol, alors on affecte 15h50 pour ce vol. C'est la raison pour laquelle la fusion se réalise et le système donne un score de *trust-worthiness* aux sources dans cette étape là.

Par ailleurs, l'article introduit une méthode pour la mise en copie de sources, *copying source* en anglais, mais leur méthode n'est pas suffisante d'après eux. Il faudrait une méthode plus robuste.

Dans cet article, on souligne les inconvénients à l'étape de pre-processing de données. Les auteurs ont récupéré les données depuis multiple sources mais ils ont rencontré un problème relatif aux conflits de données de même contenu. Même si le contenu d'un élément est le même dans deux sources différents, leurs représentations pourrait se différencier (notamment dans l'exemple de *bourse*). La qualité de résultat dépend donc trop de la pre-processing de données.

Même si l'article précédent donne un score de *worthiness* aux sources, son but principal est de tester les méthodes de fusion de données existantes. Pourtant, l'article [4] propose 2 méthodes pour estimer le *trustworthiness* d'une source. La première méthode construit un tuple (*extractor, website, predicate, pattern*) pour chaque source et introduit un ensemble de variables booléennes pour les affecter aux sites web au sujet de la justesse (0:mauvais et 1: bien). D'autre part, la deuxième méthode distingue 2 types d'erreurs: L'erreur d'extraction et l'erreur de source. Pour cela, elle introduit 2 ensembles de valeurs booléennes: L'une est pour la justesse de la source, l'autre est la justesse de l'extracteur.

Les auteurs testent leurs méthodes avec les résultats de *PageRank* car il n'y a pas de vérité de terrain, *ground truth* en anglais, sur le web pour déterminer quelles sources web sont *trust-worthiness*.

En plus de cela, l'article décide la granularité d'une source. C'est-à-dire que s'il y a trop de triplets extraits par une source, le système sépare les triplets en quelques parties pour que la qualité des résultats soient bonne.

4 Conclusion

Dans un premier temps, nous avons abordé la possibilité d'avoir une plateforme de fact-checking qui aiderait à la vérification des faits en tant qu'outil d'aide en utilisant l'approche du Crowd-sourcing, puis en trouvant des questions intéressantes à partir d'affirmation et enfin en se servant de la fouille de données. Les trois approches nécessitent d'un expert qui puisse guider le processus de l'exécution et des données structurées.

Nous avons ensuite abordé la vérification automatique de faits. Cette partie était composée de 3 pistes: la détection de faits, la vérification de faits et l'amélioration de la vérification de faits. La détection automatique de faits sans assistance humaine se réaliserait à partir d'un document ou d'un discours en temps-réel. Pour cela, la plupart des types de faits devraient être détectés et donc modélisés dans les plateformes de détection. En plus, on devrait prendre en compte de la combinaison de quelques phrases consécutives afin d'améliorer ce genre de système si la détection se réalise à partir d'un texte.

Vient aussi le problème de la vérification automatique de faits qui pourrait être résolu soit par la catégorisation des affirmations soit par l'approche de Knowledge Graph. La catégorisation des affirmations nécessite qu'une base de données soit fournie alors que l'approche de Knowledge Graph utilise des données RDF récupérées sur le Web et n'a donc pas besoin d'utiliser une base de données supplémentaire.

Finalement, nous pourrions améliorer la vérification de faits en mesurant la qualité des faits ou la qualité de la source sur laquelle le fait repose. Les travaux en cours ne sont pas suffisants et satisfaisants et il est encore nécessaire de les développer.

Grâce à notre travail sur l'état de l'art dans le domaine du Fact Checking, nous sommes convaincus qu'un système automatique de détection et de vérification de faits qui serait plus abouti est possible. De plus, l'approche par la fouille de données n'a jamais été réellement utilisé et elle semble pouvoir être un piste d'analyse intéressante pour extraire des informations et comprendre l'histoire des faits. Par ailleurs, le sujet de *provenance* pourrait être abordé afin de rendre le système plus robuste.

References

[1] Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis Mateus Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *CoRR*, abs/1501.03471, 2015.

[2] Sarah Cohen, James T. Hamilton, and Fred Turner. Computational journalism. *Commun. ACM*, 54(10):66–71, October 2011.

[3] Sarah Cohen, Chengkai Li, Jun Yang, and Cong Yu. Computational journalism: A call to arms to database researchers. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pages 148–151, 2011.

[4] Xin Luna Dong, Evgeniy Gabrilovich, Kevin Murphy, Van Dang, Wilko Horn, Camillo Lugaresi, Shaohua Sun, and Wei Zhang. Knowledge-based trust: Estimating the trustworthiness of web sources. *CoRR*, abs/1502.03519, 2015.

[5] B. Walenz et al. Finding, monitoring, and checking claims. *Computation+Journalism Symposium*, 2014.

[6] Samantha Finn, Panagiotis Takis Metaxas, and Eni Mustafaraj. Investigating rumor propagation with twittertrails. *CoRR*, abs/1411.3550, 2014.

[7] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 413–422, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[8] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Fact Checking and Analyzing the Web. In *SIGMOD - ACM International Conference on Management of Data*, New York, United States, Jun 2013.

[9] Naeemul Hassan, Bill Adair, James T. Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. The quest to automate fact-checking. 2015.

[10] X. Li, X. Luna Dong, K. Lyons, W. Meng, and D. Srivastava. Truth Finding on the Deep Web: Is the Problem Solved? *ArXiv e-prints*, March 2015.

[11] John Lieber and Adrien Coulet. Fouille de données: notes de cours [en ligne]. 2007. Disponible sur: *www.loria.fr/ coulet/material/fdd_cours.pdf* (consulté le 30.12.2015).

[12] Claudia Marinica, Fabrice Guillet, Henri Briand, and COD LINA-Equipe. Vers la fouille de règles d'association guidée par des ontologies et des schémas de règles. *Qualité des Données et des Connaissances*, 2008.

[13] Amedeo Napoli. A smooth introduction to symbolic methods for knowledge discovery. Interne, 2005.

[14] Thi K. Nguyen, Loic Cerf, Marc Plantevit, and Jean-Francois Boulicaut. Multi-dimensional Association Rules in Boolean Tensors. In *11th SIAM International Conference on Data Mining SDM'11*, pages 570–581. SIAM, April 2011.

[15] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *ArXiv e-prints*, March 2015.

[16] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *INFORMATION SYSTEMS*, 24:25–46, 1999.

[17] Baoxu Shi and Tim Weninger. Fact checking in large knowledge graphs - A discriminative predicate path mining approach. *CoRR*, abs/1510.05911, 2015.

[18] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. Toward computational fact-checking. *Proc. VLDB Endow.*, 7(7):589–600, March 2014.

[19] You Wu, Boulos Harb, Jun Yang, and Cong Yu. Efficient evaluation of object-centric exploration queries for visualization. *PVLDB*, 8(12):1752–1763, 2015.