Vérifier le respect des règlements dans le monde numérique

Michel de Rougemont
Université Paris Panthéon Assas
Institut de Recherche Fondamentale en Informatique (IRIF-CNRS)

Nous présentons la dichotomie entre *Calculer* et *Vérifier*, en particulier l'utilisation du hasard et de l'interaction pour la vérification. Un algorithme de vérification fondé sur le hasard et l'interaction comporte une probabilité d'erreur, aussi appelée risque.

Nous présentons quelques cas d'algorithmes de vérification pour s'assurer du respect du droit à l'oubli du RGPD d'une plateforme et pour la détection de messages textuels inappropriés.

1. Introduction

Le monde numérique s'est développé dès 1995 lorsque l'usage des services informatiques tels que l'email et les moteurs de recherche se sont généralisés à une grande fraction de la population mondiale. Des plateformes monopolistiques sont apparues et les Etats cherchent à réguler l'ensemble de ces services.

Pour réguler un service numérique, il est préférable de bien le comprendre et à cette fin de pouvoir décider s'il satisfait des propriétés simples. C'est le but de la vérification, par opposition aux algorithmes de calcul utilisés par les plateformes. Nous insistons sur la différence entre ces deux notions et en particulier sur la notion de *vérification approchée*, qui utilise deux concepts complémentaires : le *hasard* et l'*interaction*.

Nous présentons les *preuves interactives* comme modèle de vérification approchée en montrant que certains problèmes difficiles à calculer peuvent être faciles à vérifier dans ce sens. A titre d'exemple nous étudions une fonction d'une matrice à coefficients entiers, le *Permanent* d'une matrice, fonction difficile à calculer, mais facile à vérifier à l'aide d'une preuve interactive. Nous montrons comment la notion de preuve interactive peut être utilisée pour la vérification de l'Article 17 du RGPD, le droit à l'oubli, et répondre ainsi à la question de savoir si une plateforme numérique est en conformité avec le RGPD.

Nous étudions ensuite comment les méthodes de Machine Learning permettent des avancées dans le traitement du langage naturel, comme la traduction automatique et la classification de texte. On peut ainsi vouloir détecter des propriétés particulières de textes, comme le fait d'utiliser un langage inapproprié ou des fausses nouvelles (fake news). Nous montrons comment de simple tests permettent de vérifier de telles propriétés.

La régulation des services numériques doit pouvoir utiliser ces techniques de vérification approchée pour s'assurer que les différents règlements sont respectés. Les algorithmes de vérification deviennent alors les principaux atouts à notre disposition.

Les règlements comme les différents Articles du RGPD et d'autres règlements doivent être conçus afin qu'il existe une vérification algorithmique approchée. Dans le cas contraire, il semble difficile de pouvoir les faire respecter dans le monde numérique.

Dans la section 2, nous distinguons les algorithmes de calcul des algorithmes de vérification. Dans la section 3, nous présentons les preuves interactives et donnons dans la section 4 un exemple de son application. Nous étudions la vérification de certaines propriétés du langage naturel dans la section 5.

2. Algorithmes de calcul et de vérification

Dans le monde numérique, il y a une distinction essentielle entre *calculer* et *vérifier*. Dans le cas d'une fonction comme la multiplication de deux entiers, M(x,y)=z, un algorithme de calcul A prend x,y en entrée et trouve z en sortie, alors qu'un algorithme de vérification B prend x,y,z en entrée et donne 1 (correct) en sortie si M(x,y)=z, c'est-à dire si z est exact et donne 0 (incorrect) si z est incorrect.

> Exemple : M(50, 3)=150. L'algorithme de calcul A(50,3) trouve une valeur z. L'algorithme de vérification B(50,3, 47) donne 0 comme sortie car 47 n'est pas la valeur exacte, alors que B(50,3,150) donne 1 comme sortie car 150 est la valeur exacte. Imaginons des grandes valeurs pour x,y et z, que l'on quantifie par le nombre n de chiffres décimaux. Si n=5,
>
> M(12345, 54321)=670592745
>
> L'algorithme standard de multiplication va devoir réaliser environ n^2 additions. Si n=500, on aurait alors 250.000 additions. Pourtant, la vérification peut être plus simple, avec seulement 2000 additions si on suit l'algorithme de *la preuve par 9,* plus efficace mais approché. Il procède ainsi :
>
> - Entrée :x,y,z
> - Additionner les chiffres de x modulo 9: x'= 1+2+3+4+5 ≡6 (9), de même pour y (y'=6), et z (z'=6+7+0+5+9+2+7+4+5≡0 (9))
> - Multiplier x' et y' : x'*y' =t et prendre le module de la somme des chiffres de t modulo 9 soit t' : t=6*6=36 et t'=3+6≡0 (9).
> - Si t'=z', alors Sortie=1, sinon Sortie=0.
>
> Remarquons que ce test est plus rapide que l'algorithme de calcul, mais pas tout à fait exact. En effet, si on permute deux chiffres distincts du z correct, par exemple les deux derniers chiffres, le résultat sera alors inexact mais le test indiquera 1, car la valeur z' sera inchangée. Dans ce cas, l'algorithme de vérification est inexact et pourtant reste très utile.

A la base d'un algorithme de vérification, réside la notion de *certificat*. Comment montre-t-on qu'on est vacciné ? On montre un QR code qui est validé par un algorithme de vérification. Pourtant certains QR codes sont des faux et le test peut donc être inexact.

2.1 Preuve et certificat

Le concept de *preuve* est utilisé dans des contextes différents :

- En Mathématique, une preuve est une séquence d'expressions $\phi_1,\ldots,\phi_n$ dans un langage formel où chaque ϕ_i est une conséquence logique des expressions précédentes. Les premières formules $\phi_1\ldots\ \phi_k$ constituent l'énoncé et ϕ_n est la conclusion. Une preuve est vérifiée, par un relecteur ou par un algorithme de vérification, s'il peut garantir que la preuve est correcte, c'est-à-dire que chaque ϕ_i est une conséquence logique des $\phi_1,\ldots,\phi_{i-1}$, grâce à une déduction logique.

- En Informatique, on cherche souvent à montrer qu'un programme suit une spécification, c'est-à-dire la description de la fonction réalisée par ce programme. On produit alors une preuve mathématique comme décrit au paragraphe précédent. Lorsqu'on souhaite identifier une personne à l'aide d'une empreinte digitale ou génétique, qui constituent des *certificats*, ces procédures ne sont pas entièrement fiables et contiennent des erreurs. La probabilité d'erreur est aussi appelée le risque.

  La preuve mathématique sans erreur et la preuve approchée reflètent deux conceptions différentes de l'Informatique, que sont les méthodes formelles de l'Informatique fondamentale d'une part et les méthodes de la science des données d'autre part. Ces dernières sont proches de la physique en ce qu'elles utilisent des mesures. Les faits jouent le rôle de certificats et l'argumentation sert de *preuve*. L'algorithme de vérification va vérifier les faits et l'argumentation.

- Pour la vérification de règlements dans le monde numérique, les faits représentent les mesures possibles des plateformes et l'algorithme de vérification va décider avec une grande probabilité si la plateforme suit le règlement.

2.2 Preuves interactives

L'identification informatique la plus simple d'une personne est celle opérée par le *login* et le *mot de passe*. Cependant, ces informations peuvent être compromises et le test de validation (un mot de passe correct) ne garantit pas l'identification de la personne, du fait du risque de fausse identité.

Pour protéger ces informations confidentielles, on peut imaginer une identification où l'utilisateur ne communique pas son mot de passe mais répond à des questions aléatoires.

On peut ainsi demander quelle est la *i*-ème lettre du mot de passe pour une position aléatoire *i*. En répétant ce test plusieurs fois, on peut s'assurer que la personne connait le mot de passe sans qu'il soit transmis entièrement. Si 100 symboles sont possibles par position (lettres majuscules, minuscules, chiffres, caractères de ponctuation) et si on pose 3 questions, alors la probabilité de répondre correctement sans connaitre le mot de passe est de $(10^{-2})^3=10^{-6}$, soit une chance sur 1 million.

On utilise dans ce test deux outils essentiels *: l'interaction* en alternant question et réponse et le *hasard* car les positions *i* sont aléatoires. On peut cependant regrouper les tirages au début du processus et poser une seule question. Dans ce cas, on supprime l'interaction. Dans l'exemple qui suit, l'*interaction* est nécessaire.

3 La vérification interactive : la fonction *Permanent* d'une matrice

Soit A une matrice *(n,n),* avec *n* lignes et *n* colonnes où chaque élément $a_{i,j}$de la matrice A est un entier. La fonction *Permanent*[1] prend une matrice en entrée et calcule une valeur entière en sortie, définie par :

$$Perm\ \ (A) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

où σ est une permutation de l'ensemble {1,2,…n}. Si n=3 et $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ alors :

$$Perm\ \ (A) = 1.5.9+1.6.8+2.4.9+2.6.7+3.4.8+3.5.7=435$$

On additionne 6 termes, car le nombre de permutations de l'ensemble {1,2,3} est 6. Si *n* est grand, le nombre de termes va être de l'ordre de 2^n et donc très vite il sera trop grand. Le calcul du permanent devient alors *difficile*. Qu'en est-il de la vérification du permanent ?

Considérons la fonction :

$$Verif - Permanent(A, b) = \begin{cases} 1 & si \quad Perm\ \ (A) = b \\ 0 & si \quad Perm\ \ (A) \neq b \end{cases}$$

Cette fonction est très différente de la fonction *Permanent*, car on peut vérifier si un *Prouveur* peut nous convaincre qu'il sait calculer le *Permanent* à l'aide d'une preuve interactive qui combine les deux points essentiels :

- L'*interaction*, la possibilité de poser plusieurs questions aux prouveurs,
- Le *hasard*, la possibilité de tirer *uniformément* une valeur entière i dans un intervalle [1,…N] avec probabilité *1/N*.

[1] Cette fonction est similaire à une fonction plus classique de l'algèbre linéaire, le *déterminant*. La définition est similaire mais on introduit des signes négatifs selon la signature de chaque substitution. Le déterminant est beaucoup plus facile à calculer que le permanent.

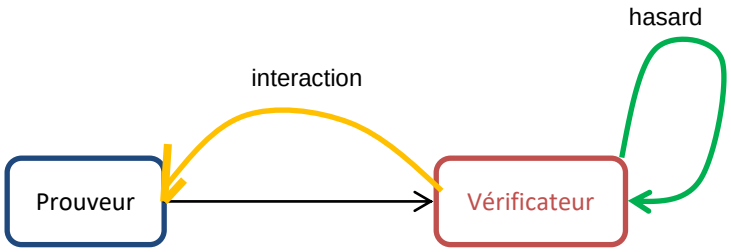

Figure 1.

Le concept de preuve interactive a été introduit dans les années 1990, voir [3], et peut se résumer dans la figure 1. Dans le schéma d'une preuve classique, un *Prouveur* transmet une preuve à un *Vérificateur*. Dans le nouveau schéma, le *Vérificateur* pose des questions au *Prouveur* en utilisant le hasard. Il va ainsi se convaincre que le Prouveur sait calculer le *Permanent.*

Prenons l'exemple d'une matrice A_n où n=100 : le *Vérificateur* demande au *Prouveur* la valeur b_n du *Permanent* de A_n, puis construit une matrice plus petite, de taille *n-1*, A_{n-1} et demande au *Prouveur* la valeur b_{n-1} du *Permanent* de A_{n-1}. Le Vérificateur réalise alors un Test simple. Il continue ainsi pour les valeurs *n-2, n-3,…1.*Pour une matrice de taille n=1, le *Permanent* est la valeur du seul coefficient. Comment est conçue la matrice A_{n-1} ? La construction dépend de A_n, b_n et d'une valeur aléatoire λ_1. Cette construction est symbolisée par la fonction f. On écrit alors $A_{n-1}= f(A_n, b_n,\lambda_1)$ pour expliciter que la construction ne dépend que de seules valeurs A_n, b_n,λ_1.

On peut ainsi représenter l'interaction par la séquence ci-dessous où les valeurs des b_n ,…, b_1 sont arbitraires. Sur la colonne gauche, le *Prouveur* trouve les valeurs b_n ,…, b_1 et les envoie au *Vérificateur*. Sur la colonne droite le *Vérificateur* construit les nouvelles matrices les renvoie au *Prouveur* et applique un Test local.

$Perm(A_n)=43512345789=b_n$	$A_{n-1}= f(A_n, b_n,\lambda_1)$
$Perm(A_{n-1})=810457839=b_{n-1}$	Test , $A_{n-2}= f(A_{n-1}, b_{n-1},\lambda_2)$
$Perm(A_{n-2})=2340057=b_{n-2}$	Test, $A_{n-3}= f(A_{n-2}, b_{n-2},\lambda_3)$
…..	….
…..	….
$Perm(A_1)=23=b_1$	Test, Oui ou Non

Si les valeurs sont correctes, tous les Tests seront valides et la sortie sera Oui. Si l'une des valeurs est incorrecte le Test le détecte avec une grande probabilité. Ainsi, à la fin de l'interaction, le Test détectera si la valeur initiale est erronée avec grande probabilité.

Le *Test de propriété*, une version plus faible d'une preuve interactive introduite dans [2] conserve le hasard et l'interaction en s'imposant uniquement des requêtes aléatoires sur l'entrée. Le Test est négatif avec grande probabilité si l'entrée est loin de satisfaire la propriété. On peut généraliser cette approche, dans [1], pour décider si deux propriétés sont similaires.

4. Vérifier le droit à l'oubli

Le Règlement Général pour la Protection des Données (RGPD) regroupe 99 Articles regroupés en 11 Chapitres. L'Article 17 du Chapitre 3 concerne le « droit à l'oubli ».

Article 17 : droit à l'effacement (droit à l'oubli). La personne concernée a le droit d'obtenir du responsable du traitement l'effacement, dans les meilleurs délais, de données à caractère personnel la concernant et le responsable du traitement a l'obligation d'effacer ces données à caractère personnel dans les meilleurs délais, lorsque l'un des motifs suivants s'applique: (6 conditions).

Les plateformes doivent respecter le RGPD et donc l'article 17. Comment s'assurer qu'elles le suivent vraiment ? Comment prouver qu'elles ne le respectent pas ?

Prenons l'exemple de la plateforme Facebook (nommée FB par la suite). Considérons l'algorithme ci-dessous, présenté sous la forme de pseudo instructions. Le résultat est OUI si la plateforme suit l'Article 17 et NON sinon.

Vérification de l'Article 17 du RGPD :

1. *Créer 2 comptes sur FB : C1 et C2*
2. *C1 et C2 « animent » leurs comptes pendant une période T*
   - *C1 Génère des « Posts » (messages) et garde une copie*
   - *C2 Génère des « Likes » des « Posts » de C1*
   - *C1 et C2 participent à des groupes*
3. *C1 annule son compte*
4. *C1 vérifie que FB lui renvoit tous ses « Posts », « Likes »,….*
5. *C2 Vérifie que les « Posts » de C1 n'apparaissent plus sur FB*
6. *Si C1 ne reçoit pas tous ses messages à l'étape 4 ou si C2 trouve des « Posts » de C1 après l'annulation de C1, alors NON, sinon OUI.*

L'activité d'animation des deux comptes C1 et C2 peut être générée par un algorithme et donc un Bot, qui va utiliser le *hasard*. L'*interaction* est présente entre C1 et C2, puis entre C1 et la plateforme lorsque C1 annule son compte, puis entre C2 et la plateforme lorsque C2 vérifie si les « Posts » de C1 sont toujours présents.

On illustre donc bien comment utiliser ces deux ressources complémentaires, *l'interaction* et le *hasard*. Si la plateforme suit l'Article 17, l'algorithme répond OUI, et il

reste à montrer que si la plateforme ne suit pas l'Article 17, l'algorithme répond NON avec grande probabilité. Cette étape technique dépasse le cadre de cet article.

5. Classification en langage naturel

Les textes écrits dans différentes langues sont classifiés à l'aide de différentes techniques de *Machine Learning*. Un classificateur prend un texte en entrée et produit une classe comme sortie (par exemple Business, Sports, Politique). Les méthodes [4,5,6] ont permis des avancées significatives au cours des dernières années. Ce sont des méthodes de calcul parmi de nombreuses possibles qui reposent sur plusieurs principes simples :

- Associer un vecteur de petite dimension à chaque mot de manière à ce que les mots dont le sens est proche soient proches comme vecteurs. Le système Word2Vec [4] réalise cette construction, à partir de techniques d'apprentissage en lisant un grand ensemble de textes, par exemple tout Wikipédia dans une langue L fixée à l'avance. Soit v_i le vecteur associé au mot i.
- Associer un vecteur à chaque phrase d'un texte à partir des vecteurs de chacun des mots de manière à ce que des phrases dont le sens est proche soient aussi des vecteurs proches. Les techniques d'*attention* [6] sont la base des techniques de *Transformers* qui réalisent cette opération, comme celles présentées dans [5].

Les perspectives d'application du traitement du langage naturel sont nombreuses. La traduction automatique d'une langue vers plusieurs langues, la classification de textes, la détection d'expressions inappropriées, et de fausses nouvelles sont des exemples importants.

Considérons le problème de la simple détection d'un langage inapproprié. On a alors une seule propriété, ou deux classes opposées (par exemple le texte est-il inapproprié ou non) et on est confronté à un problème de vérification, à savoir le texte a-t-il ou non cette propriété ? Vérifier une propriété est plus simple que classifier un texte avec la méthode qui suit. Elle sera en particulier utile pour détecter des messages inappropriés sur les plateformes des réseaux sociaux.

Pour une phrase donnée *p* soit *l'attention du mot i de p pour la langue L,* notée *A(p,i,L)*, la valeur définie comme suit *:*

$$A(p,i,L) = \sum_{j \ dans \ p} v_i . v_j$$

Si la phrase p contient 10 mots, on a une somme de 10 expressions $v_i . v_j$ qui utilise les vecteurs v_i d'une langue L, obtenus par apprentissage sur un grand corpus. Soit l'indice de la phrase *p* pour L, noté $I(p,L)$, défini par la formule :

$$I(p,L) = \sum_{i\ dans\ p} A(p,i,L)$$

L'indice de *p* représente le poids ou l'énergie de la phrase *p* pour la langue *L*.

Soit v'_i un vecteur similaire pour une langue L', construite comme une série d'expressions inappropriées, obtenues également par apprentissage. On peut alors comparer $I(p,L)$ et $I(p,L')$ et obtenir un Test simple.

Algorithme de vérification d'un texte pour une langue L':

1. *Echantillonner le texte : soit les phrases p_1, …, p_k*
2. *Pour chaque phrase p_1, …, p_k*
   - *Calculer l'indice $I(p_i, L)$*
   - *Calculer l'indice $I(p_i, L')$*
3. *Calculer les moyennes des indices pour p_1, …, p_k*
   - *Soit l'indice $I(L) = \sum_{i=1\ldots k} I(p_i, L)/k$*
   - *Soit l'indice $I(L') = \sum_{i=1\ldots k} I(p_i, L')/k$*
4. *Si $I(L') > 2.I(L)$ alors OUI, sinon NON*

Ce Test ne présente aucune *interaction*, mais utilise uniquement le *hasard* lors de l'échantillonnage. Si le texte est court, comme un tweet, on considère toutes les phrases, sans échantillonner et le Test devient *déterministe*. A l'instruction 4, on teste si l'indice moyen pour L', est supérieur à deux fois l'indice moyen pour L. On peut bien sûr remplacer le 2 par une constante plus petite.

6. Conclusion

Nous avons présenté la dichotomie entre les algorithmes de calcul et les algorithmes de vérification, qui semblent être de bons outils pour la régulation des services numériques.

La vérification peut utiliser deux possibilités complémentaires : le *hasard* et l'*interaction*. Dans le cas du *hasard*, la vérification sera correcte avec grande probabilité et ne sera pas parfaite. On parle alors de vérification approchée. Nous avons donné quelques exemples classiques comme la vérification de l'identité et celle du *Permanent* d'une matrice, une fonction difficile à calculer mais facile à vérifier dans ce sens.

Nous avons pris comme exemple le droit à l'oubli, Article 17 du RGPD, et imaginé un algorithme de vérification approché pour une plateforme comme Facebook. Nous avons ensuite présenté une méthode très simple de détection de langage inapproprié, en utilisant les outils d'analyse du langage naturel.

Références

[1] Fischer, E. and Magniez, F. and de Rougemont, M.
Approximate Satisfiability and Equivalence (2010)
SIAM Journal of Computing (39), pages 421-430

[2] Goldreich, O. and Goldwasser, S. and Ron, D.
Property testing and its connection to learning and approximation (1998)
Journal of the ACM (45), pages 653—750.

[3] Goldwasser, S. and Micali, S. and Rackoff, C.
The knowledge complexity of interactive proof systems (1989)
SIAM Journal of Computing (*18),* pages 186-208

[4] Tomas Mikolov and Kai Chen and Greg Corrado and Jeffrey Dean
Efficient Estimation of Word Representations in Vector Space (2013)
CoRR, abs/1301.3781

[5] N. Reimers and I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks (2019),arXiv 1908.10084

[6] Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin, Attention Is All You Need (2017), arXiv 1706.03762