Le temps et le numérique

Le conseil que l'on donne souvent face à une difficulté est de laisser le temps au temps. Ce conseil est donné par Miguel de Cervantes dans son ouvrage la Gitanilla. L'auteur espagnol du 16e siècle s'exprime ainsi : **« Se dará tiempo al tiempo, que suele dar dulce salida a muchas amargas dificultades. ». Une traduction en français de cette phrase est la suivante : « On donnera du temps au temps, qui la plupart du temps donne une douce issue a beaucoup d'amères difficultés. ».**

Source : site Wikipédia. Domaine public.
Portrait imaginaire de Miguel de Cervantes

Le monde numérique qui nous entoure est construit sur des technologies qui produisent les effets attendus dans des durées de plus en plus courtes. Aussi le conseil de Cervantès de donner du temps au temps est certainement largement marginalisé au moment d'utiliser des outils numériques. Cette quasi instantanéité est une qualité majeure du numérique pour fournir par exemple des réponses à une requête sur un moteur de recherche ou pour utiliser un outil numérique pour communiquer.

Il est utile de savoir que le temps est une composante parfaitement intégrée par le numérique. C'est ce que nous vous proposons de découvrir au travers de ces deux articles dont le premier montre le module du logiciel Python entièrement réservée au temps et le second nous invite à réfléchir à la notion de temps dans le domaine de l'intelligence artificielle.

L'epoch du logiciel Python

par Jean-Alain Roddier, IA-IPR de mathématiques

Le logiciel Python est le logiciel libre de programmation utilisé à la fois dans l'enseignement secondaire à partir de la classe de Seconde mais aussi dans le monde de l'entreprise. Ce logiciel est arrivé à faire sens en tant qu'outil de programmation car une sorte de communauté Python s'est construite durant ces dix dernières années. Cette communauté de programmeurs de tout âge arrive aujourd'hui à échanger et à s'entraider autour du développement de programmes en Python.

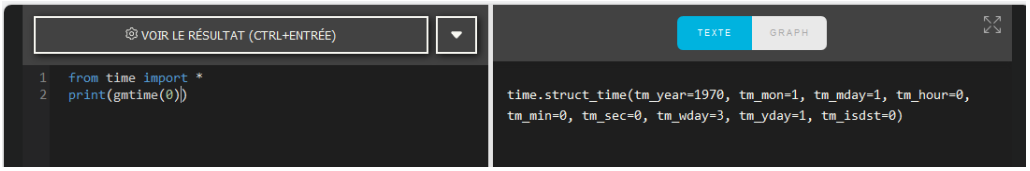
Le logiciel Python est disponible aujourd'hui dans des versions utilisables en ligne. Pour illustrer cet article, nous allons prendre la version proposée par le site lemanuelscolaire.fr. Ce site donne accès à une version de Python qui ne nécessite aucune installation sur poste informatique dudit logiciel. https://www.lelivrescolaire.fr/outils/console-python

Le logiciel Python contient un module (bibliothèque de programmes) uniquement consacré au temps, c'est dire l'importance de la notion de temps au moment de programmer.

Qu'est-ce que l'epoch ?

Pour mesurer une durée, il s'agit avant tout de fixer un instant de départ du temps (instant où on lance le chronomètre). **Cet instant de départ s'appelle l'epoch**. Ce nom commun epoch est réservé aux initiés mais au moment de parler du temps, il est incontournable de savoir de quoi l'on parle. Pour de très nombreux systèmes d'exploitation, l'epoch est le 1er janvier 1970 à 0 heures, 0 minutes et 0 secondes (heure de Greenwich UTC+0 noté aussi GMT). Si l'on veut en avoir une image plus festive, c'est l'instant où les personnes situées sur la planète dans le fuseau horaire UTC+0 ont fêté le jour de l'an pour passer de 1969 à 1970. Cette heure et cette date de l'epoch sont données en tapant sur python les lignes d'instructions suivantes :

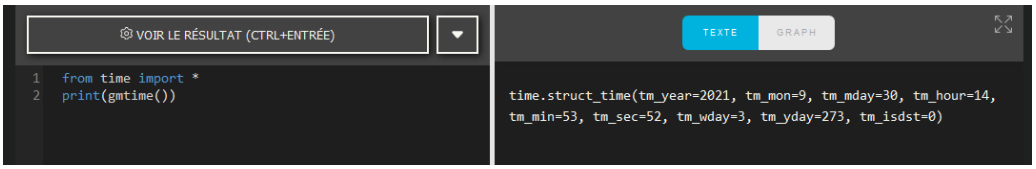
from time import *	permet d'ouvrir la bibliothèque time et rend accessible un ensemble de programmes)
print(gmtime(0))	renvoie l'epoch du logiciel Python



Une parfaite connaissance de la date et de l'heure.

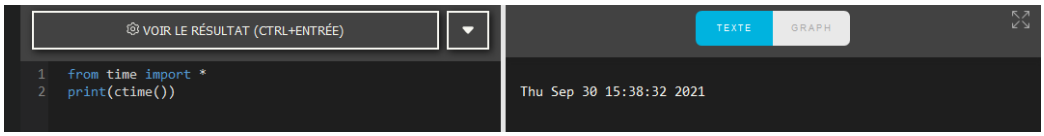
Nous allons voir à présent que le logiciel a une parfaite connaissance de la date et de l'heure.

On peut ainsi demander au logiciel d'identifier l'heure et le jour auxquels on est en train d'utiliser l'ordinateur. Pour ce faire, nous utilisons l'instruction gmtime() qui renvoie les valeurs de 9 variables :



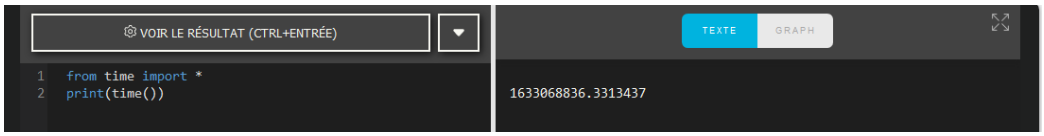
La première question que l'on se pose est de savoir si le logiciel renvoie bien les bonnes valeurs autrement dit si celles-ci correspondent à ce qu'affiche notre montre. On doit faire attention à bien prendre en considération d'une part le décalage horaire qui en heure d'été correspond à UTC+2 et d'autre part les définitions de trois de ces neuf variables :
tm_wday désigne le nombre de jours écoulés compté à partir de lundi ;
tm_yday est le nombre de jours écoulés depuis le 1er janvier de l'année en cours ;
tm_isdst fait référence à heure d'été/heure d'hiver (la valeur est 0 si l'heure d'été n'est pas appliquée).

Pour obtenir le jour de la semaine, la date et l'heure, on peut lancer l'instruction ctime().



Une connaissance précise du temps écoulé depuis epoch

Comme nous venons de le voir le logiciel connaît parfaitement la date et l'heure ce qui correspond pour l'instant à une précision d'une seconde. Nous allons voir à présent que le logiciel est bien plus précis et que l'instruction time() permet de connaître le temps écoulé depuis epoch au 10 millionième de seconde près.



L'écran ci-dessus s'interprète ainsi : à l'instant où le logiciel a fini de traiter la deuxième instruction, il s'est écoulé 1633068836,3313437 secondes depuis epoch.

A quoi sert une telle précision ?

La précision fournie par la fonction time() permet entre autres de mesurer la durée d'exécution d'un programme. Pour rendre les choses plus concrète, nous allons prendre un exemple celui d'un programme qui permet d'écrire 1000 fois le début de la phrase de Cervantes : « *Se dará tiempo al tiempo* ». On va ainsi considérer la suite d'instruction ci-dessous :

from time import *	On ouvre la bibliothèque time
début=time()	On met dans début le temps écoulé depuis epoch
for i in range(999) :	On fait varier une variable i de 0 à 999
Print(« Se dara tiempo al tiempo »)	Pour chaque valeur de i, on fait afficher le début de ladite phrase
fin=time()	On met dans fin le temps écoulé depuis epoch
print(fin-début)	On fait afficher la différence entre début et fin.

N.B. : Il faut rester vigilant concernant la véracité des décimales fournies par le logiciel car des erreurs peuvent être bien présentes du fait de l'écriture en binaire des nombres décimaux. Lancer par exemple la séquence : print(0.1+0.2) et vous pourrez par exemple constater que la réponse fournie n'est pas 0.3.

Conclusion

La durée d'exécution d'un programme est une donnée dont les développeurs tiennent compte pour arriver par exemple à traiter le maximum de données en un temps limité. C'est ainsi un enjeu d'envergure pour le numérique d'aller de plus en plus vite. Cette prise en compte du temps est une composante dont l'utilisateur ne voit que les effets. Il est utile que la formation au numérique participe à rendre le numérique plus transparent, cela a été ici notre intention.

Le temps passe-t-il pour l'intelligence artificielle ?
par Nazim Fatès
Chargé de recherche *INRIA* Nancy-Grand Est

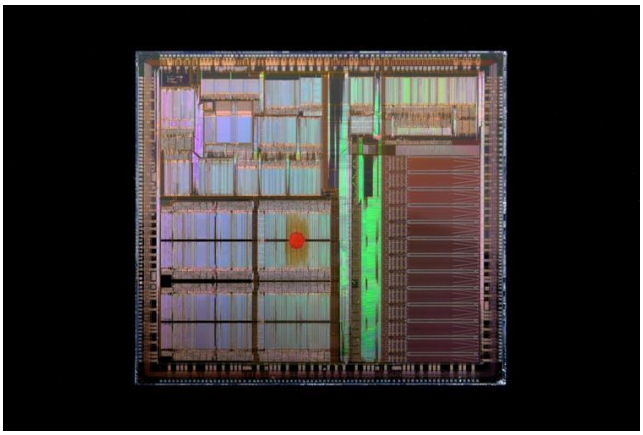
Les systèmes d'intelligence artificielle sont des systèmes informatiques qui sont le plus souvent dotés de la possibilité d'évoluer, de s'adapter et de s'automodifier. Leurs concepteurs cherchent à les rendre aussi autonomes que possible et ils en viennent souvent à se demander dans quelle mesure de tels systèmes pourraient acquérir la notion du temps.

Le problème central est celui de l'interprétation des données que l'on fournit à une machine pour qu'elle apprenne : les données de l'expérience ont besoin du temps pour être interprétées et, réciproquement, le temps a besoin de l'expérience pour prendre sa consistance et permettre l'interprétation des données. Il y a donc une intrication. On peut bien sûr apprendre à une machine toute sorte de choses, comme distinguer des tumeurs bénignes de tumeurs malignes sur des photos médicales, mais comment un robot pourrait-il se construire une notion de « temps », avec toute la richesse que ce mot représente ? Par exemple, dans le contexte d'une interaction avec des êtres humains, comment faire en sorte qu'un robot sache de lui-même s'il n'est pas trop rapide ou trop lent ? Comment parviendrait-il à se rendre compte que quelque chose a brusquement changé dans le comportement de son interlocuteur ?

Le temps des robots est le temps des microprocesseurs

À ce jour, l'ensemble des systèmes informatiques fonctionne sur les bases algorithmiques posées par Alan Turing en 1936. Alan Turing partit de l'hypothèse que toute méthode systématique de résolution d'un problème, c'est-à-dire tout algorithme, peut être traduite dans un langage qui s'adresse à une machine élémentaire réalisant des opérations de lecture-écriture sur un ruban infini.

Les systèmes informatiques dont nous disposons n'opèrent pas exactement sur ce type de machine, mais on admet généralement un principe d'équivalence : tout ce qui peut être réalisé par une machine donnée peut également être réalisé par cette machine de Turing, dite « universelle ».



Un chip de microprocesseur. Laura Ockel/Unsplash, CC BY

Ce point est particulièrement important pour comprendre les évolutions temporelles des systèmes d'intelligence artificielle. En effet, ceux-ci utilisent des calculs « parallèles » : au lieu de faire une opération après l'autre, ces systèmes peuvent, comme dans un cerveau, faire interagir des milliers de composants simultanément. On parle souvent à propos de telles

architectures de « connexionnisme » : il ne s'agit pas seulement comme dans le cas du parallélisme classique de faire interagir plusieurs systèmes en même temps, mais de parvenir à coordonner une myriade d'unités de calcul, et ce sans unité centrale.

Dans ce contexte, le principe d'équivalence énoncé par Turing tient encore : une architecture en réseau peut accélérer les calculs, mais ne peut jamais permettre de faire ce qui est hors de portée pour une machine séquentielle. En ce qui concerne le temps d'exécution des algorithmes, cela signifie que si j'ai une machine avec des millions de neurones formels qui changent d'état en parallèle, j'aurais probablement la possibilité d'effectuer des algorithmes de manière plus rapide, mais le temps intrinsèque de la machine sera toujours donné par le temps des horloges des microprocesseurs qui cadencent cette machine. Il existe plusieurs dispositifs de calcul non classiques, tels que les ordinateurs quantiques ou les puces dites neuromorphiques : certes, leur programmation oblige à penser de façon différente et sont la source de nombreuses promesses pour repousser les frontières du calcul, cependant ils n'échappent nullement au principe d'équivalence de Turing et aux limites que cette équivalence impose.

Un système d'intelligence artificielle reste donc conditionné dans son rapport au temps par sa structure algorithmique discrète, laquelle décrit l'évolution des systèmes pas à pas. Le temps informatique est donc toujours mesuré comme un nombre d'étapes, que celles-ci soient parallèles ou séquentielles.

Quelles sont les conséquences de telles limitations ?

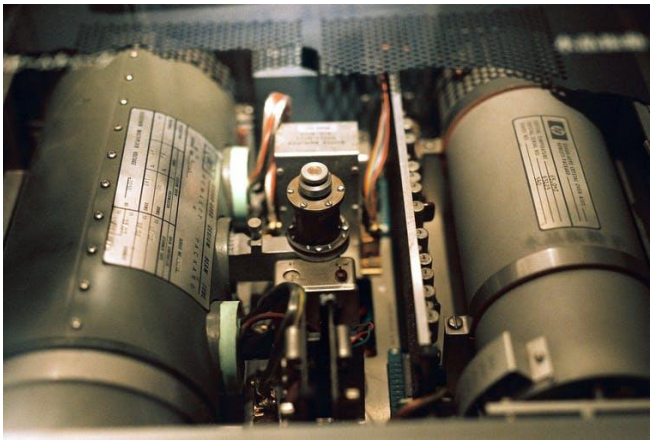
Il y a une inhomogénéité fondamentale entre le temps des êtres humains et celui des machines. Il faut garder à l'esprit que n'importe quel ordinateur, téléphone, ou même n'importe quelle puce qui se trouve dans une machine à laver effectue des milliards d'opérations par seconde. En d'autres termes, l'échelle avec laquelle les cadences des microprocesseurs sont mesurées est le gigahertz. Si l'on pouvait se placer du point de vue des robots, nous verrions les êtres humains comme des lourdauds qui pensent et se meuvent à une vitesse phénoménalement lente. On peut faire une analogie avec la manière dont les plantes évoluent pour nous. Les robots seraient donc amenés à se brider considérablement pour « s'abaisser » à notre rythme !

D'ailleurs, ces problèmes ont été perçus dès le début de la réflexion sur la question de l'intelligence artificielle. Alan Turing, par exemple, dans son article de 1950, demande à ce que la machine qui remplace un être humain jouant au jeu de l'imitation marque un temps d'arrêt artificiel avant de donner le résultat d'une multiplication, sans quoi elle serait immédiatement démasquée. De tels délais sont aujourd'hui utilisés pour rendre les conversations des « assistants vocaux » plus naturels.

La science-fiction a aussi souvent exploité le filon de l'incommensurabilité du temps humain et du temps des machines. Par exemple, dans le film *Her* de Spike Jonze (2013), le protagoniste est séduit par son système d'exploitation et finit par tomber amoureux d'« elle ». Néanmoins, au plus fort de leur liaison (platonique), elle lui avoue que, pendant la durée de leur conversation intime, elle a pu lire plusieurs milliers de livres et converser avec plusieurs centaines d'autres personnes.

Le roman d'Antoine Bello *Ada* met en scène une créature virtuelle chargée d'écrire des romans à l'eau de rose et un inspecteur qui cherche à la retrouver après qu'elle s'échappe de l'atelier de ses créateurs. Ada sait jouer avec les sentiments de l'inspecteur et elle a la fâcheuse tendance à effectuer des recherches sur des éléments de sa vie en même temps qu'ils discutent. Quant à sa collègue, Jessica, celle-ci est programmée pour écrire des biographies

personnalisées avec la faculté de traiter des dizaines de milliers de clients en parallèle… L'imaginaire de l'intelligence artificielle nous rappelle que les créatures artificielles manquent cruellement d'un ici et d'un maintenant pour pouvoir être considérées pleinement comme autre chose que des objets.



L'horloge atomique d'Hewlett-Packard,
qui définissait le temps sur le fuseau horaire japonais. halfrain/Flickr, CC BY-NC

Les chercheurs qui essaient de donner aux machines la possibilité d'interpréter le langage humain font aussi face à des défis colossaux. Saisir la temporalité reste ce qu'il y a de plus difficile. Une simple phrase comme « Maintenant, ça suffit ! », qu'un enfant comprend immédiatement, reste une énigme pour des systèmes informatiques, car que signifie ce « maintenant » ? Certainement pas la même chose que dans « Maintenant, il est temps de passer à table ». Chacun comprend que seule une expérience de la vie permet de saisir les nuances de la langue et que tout ne se ramène pas à des « faits » que l'on peut encoder dans des systèmes informatiques. En particulier, notre propre perception du temps qui passe s'inscrit dans une rythmicité journalière, qui s'inscrit dans une rythmicité plus longue (le mois, l'année, etc.), qui elle-même s'inscrit dans le chemin d'une vie, chemin qui prend son sens dans son inscription dans une histoire plus longue, voire dans un rapport à un temps non mesurable comme le montrent les mythes de toutes les civilisations.

Le véritable danger de l'intelligence artificielle, en voulant sans cesse tout accélérer, ne serait-il pas d'occulter cette dimension fondamentale de l'être humain à savoir, non pas seulement que les choses *prennent* du temps, mais aussi que la maturation de toute bonne chose demande un temps incompressible ? Les blés ont besoin de temps pour mûrir et le pain a besoin d'un temps pour cuire. Si un jour les robots comprennent cela, on pourra dire qu'ils seront devenus alors véritablement… « humains ».

Ce texte est disponible en ligne sur le site THE CONVERSATION à l'adresse suivante : https://theconversation.com/le-temps-passe-t-il-pour-lintelligence-artificielle-162291.