Traitement automatique du langage naturel

Václav Gregor

28 février 2024

Table des matières

Résumé

Ces notes présentent un regarde épistémologique sur le domaine du traitement automatique du langage naturel. Elles parlent brièvement de son histoire, passant ensuite sur l'état du domaine actuel. Ici on regarde de plus près les techniques de l'apprentissage automatique et du deep learning utilisées dans ce domaine.

Chapitre 1

Introduction

Nous allons en premier regarder l'histoire de ce domaine, où nous découvrirons les trois principales approches dans le traitement automatique du langage naturel (TALN en abrégé) : symbolique, statistique et neuronale. Il est impossible pour moi de donner un résumé complet de l'histoire de ce domaine, ainsi que de décrire chaque approche parfaitement. J'ai donc pris la décision de surtout utiliser des exemples de programmes et d'expériences spécifiques pour montrer comment le domaine a évolué.

Les insuffisances des deux premières approches nous emmèneront vers le TALN neuronal, c-à-d des techniques utilisant l'apprentissage automatique (ML en abrégé pour "machine learning") ou le deep learning (DL en abrégé). On regardera les tâches les plus courantes dans le TALN et comment on peut utiliser des modèles de ML et de DL pour les résoudre. On répondra à la question : "Pourquoi l'approche neuronale est la plus efficace ?" et on discutera des applications actuelles.

On passera ensuite sur un exemple concret : Word2Vec, qui est une technique de ML légère et puissante en même temps, permettant de transformer des mots en vecteurs. On expliquera son fonctionnement et on regardera ces applications - surtout dans les tâches du TALN qui concernent la sémantique des mots.

Finalement, on verra et essayera d'expliquer les techniques actuelles - celles utilisant le deep learning. On regardera aussi les problèmes actuels du TALN et son possible développement futur. [1] [2] [3] [4] [**?**] [**?**] [**?**]

Chapitre 2

Brève histoire du TALN

Approche symbolique

l'expérience Georgetown-IBM

Le domaine a débuté dans les années 1950. On peut dire que le premier résultat marquant et connu était l'expérience Georgetown-IBM en 1954. Il s'agissait d'une démonstration de traduction automatique du russe vers l'anglais, comme le contexte historique le voulait.

Le programme contenait 6 règles de grammaire, 250 éléments lexicaux dans son vocabulaire (thèmes et désinences des mots), et était capable de traduire 60 phrases. Les phrases parlaient surtout de la chimie, les mathématiques, la communication, la métallurgie et les affaires militaires. Il est essentiel de noter que les phrases ont été choisies soigneusement par les auteur du programme. Certaines règles et opérations du programme était spécifiques à un nombre limité de mots et de phrases d'entrée. Chaque mot russe du vocabulaire correspond à un ou deux équivalents anglais. En plus, chaque mot aurait 3 codes numériques lui associés, qui déterminent la règle de grammaire à utiliser pour produire la sortie. Si cette description nous rappelle la notion de "hardcoding", ce n'est pas par hasard. Le programme était essentiellement un dictionnaire assez limité, qui cherchait la traduction correspondante pour chaque mot russe, et qui appliquer l'une des 6 règles pour rendre la phrase anglaise à la sortie plus correcte. Regardons ceci sur un exemple.

La phrase russe (en alphabet latin)
Vyelyichyina ugla opryedyelyayetsya otnoshyenyiyem dlyini dugi k radyiusu.
Traduction française
L'ampleur de l'angle est déterminée par la relation entre la longueur de l'arc et le rayon.
Traduction anglaise obtenue par le programme
Magnitude of angle is determined by the relation of length of arc to radius.

Regardons maintenant les informations que le programme avait à sa

disposition pour traduire cette phrase.

entrée russe	équivalents anglais	code 1	code 2	code 3	règle
vyelyichyina	magnitude	***	***	**	6
ugl-	coal, angle	121	***	25	2
-a	of	131	222	25	2
opryedyelyayetsya	is determined	***	***	**	6
otnoshyenyi-	relation, the relation	151	***	**	5
-yem	by	131	***	**	3
dlyin-	length	***	***	**	6
-i	of	131	***	25	3
dug-	arc	***	***	**	6
-yi	of	131	***	25	3
k	to, for	121	***	23	2
radyius-	radius	***	221	**	6
-u	to	131	***	**	3

TABLE 2.1 – Example Table

Le premier mot (vyelyichyina) n'a qu'un seul équivalent anglais (magnitude) et son premier code étant vide (***) infère la règle 6 : la traduction est simplement copiée à la sortie.
sortie partielle : magnitude

Le deuxième mot (ugla) est séparé en thème (ugl-) et désinence (-a). Le thème (ugl-) appelle la règle 2 avec son code 1 (121). La règle 2 va chercher à trouver 221 ou 222 comme le code 1 de la prochaine entrée (-a). On trouve 222 pour le code 1 de l'entrée "-a", et donc on choisi le deuxième équivalent de "ugl-" pour la sortie (angle).
sortie partielle : magnitude angle

L'entrée suivante (-a) appelle la règle 3 avec son code 1 (131). Cette règle va regarder si le code 2 de l'entrée précédente est égal à 23. Comme ce code est vide (***), on sélectionne le seul équivalent de "-a" et on inverse l'ordre des deux mots (ce qui produit "of angle").
sortie partielle : magnitude of angle

Le reste de la traduction serait trouvé similairement, on va donc s'arrêter ici. Je pense que ce petit exemple est suffisant pour illustrer le fait que le programme ne fait que suivre bêtement les règles "hardcodées" par les chercheurs. Avant d'être programmé, ce processus qu'on vient d'essayer a été testé par des personnes avec aucune connaissance en russe, pour vérifier qu'il produit des résultats correctes sur les phrases données.

En gros, la valeur de l'expérience Georgetown-IBM repose dans l'analyse de la grammaire russe et dans l'invention des 6 règles qui permettent de traduire les phrases prédéterminées sans aucune réflexion. Cette tâche a ensuite été passé à un ordinateur.

Ce qui est intéressant est que cette expérience était vue comme un énorme succès. Voici quelques extraits d'article des journaux américains contemporains pour illustrer ce point.

> It is expected by IBM and Georgetown University, which collaborated on this project, that within a few years there will be a number of "brains" translating all languages with equal aplomb and dispatch. (Kenny, Christian Science Monitor)
>
> The girl who operated 701 did not understand a word of Soviet speech and yet more than 60 Soviet sentences were given to the "brain" which translated smoothly at the rate of about $2\frac{1}{2}$ lines a second. (Kenny, Christian Science Monitor)
>
> The "brain" didn't even strain its superlative versatility and flicked out its interpretation with a nonchalant attitude of assumed intellectual achievement. (Kenny, Christian Science Monitor)

Il faut remarquer que dans les années 1950, les ordinateurs étaient toujours vu comme des machines, ou mêmes cerveaux mystérieux par la plupart des gens. Ceci a permis à cette démonstration d'arriver à son but : attirer l'attention du publique et du gouvernement américain et obtenir des financement pour la recherche future dans ce domaine.

ELIZA

Introduction

En 1966, ELIZA est créé par Joseph Weizenbaum - un programme pour explorer l'interaction entre les humains et les machines. ELIZA est capable d'utiliser les règles définies dans un script externe pour simuler une conversation. Le programme repose sur l'application de pattern-matching et d'une méthodologie de substitution, qui seront expliqués sur un exemple. Son script le plus célèbre : DOCTOR, simule un psychothérapeute de l'école Rogerienne (de "Rogerian school" en anglais). Ce qui est important à retenir est que dans cette approche psychothérapeutique, le psychologue répète souvent les paroles du patient à ce dernier. Ce script permet donc à ELIZA de garder l'illusion d'une conversation avec un humain, sans savoir rien sur le monde réel. Citons Weizenbaum pour expliquer ce point :

> This mode of conversation was chosen because the psychiatric interview is one of the few examples of categorized dyadic natural language communication in which one of the participating pair is free to assume the pose of knowing almost nothing of the real world. If, for example, one were to tell a psychiatrist "I went for a long boat ride" and he responded "Tell me about boats", one would not assume that he knew nothing about boats, but that he had some purpose in so directing the subsequent conversation.

> It is important to note that this assumption is one made by the speaker. (Joseph Weizenbaum, ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine, 1966)

Fonctionnement général

Le fonctionnement général de ELIZA peut être décrit ainsi : Le texte entré par l'utilisateur est inspecté pour trouver l'un des "mots clés". Si un mot clé est trouvé, l'entrée est transformée en appliquant une réglé associée avec ce mot clé. Si aucun mot clé est trouvé, une réponse générale ne dépendant de l'entrée est formulée, ou bien une transformation déjà utilisée avant est appliquée. Les mots clé et leur transformations sont défini dans le script externe. ELIZA peut donc travailler avec n'importe quel script, et c'est ce dernier qui détermine "sa personnalité".

Weizenbaum résume ceci en 5 problèmes techniques fondamentaux :

- L'identification du mot clé le plus important, car il peut y en avoir plusieurs dans une seule phrase.
- L'identification d'un contexte minimal dans lequel le mot clé apparaît. Par exemple, si le mot clé est "you", est-ce qu'il est suivi par "are" ? Dans ce cas, une assertion sur ELIZA est probablement faite par l'utilisateur.
- Le choix d'une transformation appropriée et l'exécution de cette transformation.
- Un mécanisme qui permettra à ELIZA de répondre d'une façon "intelligente" si aucun mot clé est trouvé.
- Un mécanisme pour faciliter l'édition et extension d'un script. Ce point nous intéressera peu, car il n'est pas essentiel pour comprendre le fonctionnement de ELIZA.

Les transformations

Le principe des transformation est expliqué très bien par Weizenbaum dans son article originale : Considérons la phrase "Je suis très malheureux". Supposons qu'un étranger (pourquoi pas un Slovaque) avec un niveau de français limité a entendu cette phrase mais n'a compris que le début de la phrase : "Je suis". Le Slovaque a quand-même retenu le reste de la phrase, mais il ne sait pas ce que ça veut dire. Souhaitant répondre à la personne, il remplace "Je suis" par "Ca fait combien de temps que tu es" et ensuite il répète le reste de la phrase originale pour formuler sa réponse : "Ca fait combien de temps que tu es malheureux ?". Le Slovaque a appliqué un certain modèle à la phrase originale, qui l'a séparée en deux parties : "Je suis" et "très malheureux". Il a ensuite utiliser une transformation qui lui permet de répondre même s'il ne comprend pas la phrase entière. Cette transformation

lui dit qu'à toute phrase de la forme "Je suis BLABLA", il est possible de répondre avec "Ca fait combien de temps que tu es BLABLA", sans savoir ce que le BLABLA veut dire. Voilà ce que fait ELIZA.

Décomposition et réassamblage

Plus formellement, considérons la phrase "It seems that you hate me". On l'a décompose ainsi en quatre parties :

(1)It seems that (2)you (3)hate (4)me

Supposons que l'étranger ne comprend que les parties 2 et 4 de cette phrase. Une façon générale pour lui de répondre pourrait être "What makes you think I hate you ?". C-à-d il jette la partie (1), il traduit les parties qu'il a compris - "you" devient "I" et "me" devient "you" et finalement il ajoute une phrase de base "What makes you think" devant tout ça.

On peut alors représenter le modèle de decomposition que l'étranger a utilisé plus formellement

(0 YOU 0 ME)

où le 0 correspond à un nombre quelconque de mots dans la phrase originale. Voici la règle de réassamblage correspondante

(WHAT MAKES YOU THINK I 3 YOU)

où le 3 correspond à la troisième partie de la décomposition de la phrase originale - ce qui se trouve entre les seuls mots compris "you" et "me".

Si on prend la phrase "It seems that you hate", la règle de décomposition (0 YOU 0 ME) échoue, comme on n'arrive pas à trouver le mot "ME". On voit donc le besoin d'avoir plusieurs décompositions pour un seul mot clé, qui vont être testées une par une sur la phrase d'entrée. En plus, on aura plusieurs réassamblages possibles pour chaque décomposition. On peut alors représenter les données correspondantes à un mot clé ainsi :

$$\begin{aligned}\{K : &[D_1, R_{1,1}, R_{1,2} \dots R_{1,m_1}], \\ &[D_2, R_{2,1}, R_{2,2} \dots R_{2,m_2}], \\ &\dots \\ &[D_n, R_{n,1}, R_{n,2} \dots R_{n,m_n}]\}\end{aligned}$$

où à un mot clé K, on associe n règles de décomposition $D_1 \dots D_n$, et à chaque règle de décomposition D_i on associe m_i règles de réassamblage $R_1 \dots R_{m_i}$. Un script pour ELIZA n'est alors qu'une liste de telles structures. Un dictionnaire qui a comme clés les mots clés du script est construit une

fois que le script est chargé par ELIZA. En plus, les clés du dictionnaire sont haché pour permettre à ELIZA de déterminer rapidement si un mot lu est un mot clé ou pas, car la plupart des mots que ELIZA va lire ne sont pas des mot clés. Je pense que l'idée principale derrière ELIZA, ainsi que la façon dont Weizenbaum l'a implémenté sont maintenant clairs. Regardons d'autres mécanismes qui sont présents dans ELIZA.

Substitution

On remarque aussi que dans l'exemple au-dessus, on a substitué le mot "you" à l'entrée par "I" à la sortie. De telles substitutions sont aussi définies dans la liste d'un mot clé donné.

exemple.

Classement des mots clés

ELIZA contient aussi un mécanisme pour classer les mots clé par importance. D'où le besoin pour ceci ?

exemple.

Pendant le scan de la phrase à l'entrée, ELIZA utilise une autre liste pour tenir compte des mots clé rencontrés, et de le garder ordonnés par leur importance.

Dernière réassamblage utilisé

Lorsqu'une règle de réassamblage correspondante à une décomposition est utilisée, son indice va être sauvegardée. Lors de la prochaine utilisation de cette décomposition, l'indice va permettre à ELIZA d'utiliser une autre règle de réassamblage. Elle va utiliser chaque réassamblage avant de revenir sur un qu'on a déjà vu. Ceci rend les réponses de ELIZA plus riches.

Mémoire

Un mécanisme très simple mais qui pourtant produit des résultats impressionnants est celui de la mémoire. Il permet à ELIZA de répondre à l'utilisateur même si aucun mot clé n'est trouvé dans l'entrée. La réponse va donc faire référence à quelque chose que l'utilisateur a dit ultérieurement. Considérons la structure suivante :

(MEMORY MY
(0 YOUR 0 = LETS DISCUSS FURTHER WHY YOUR 3)
(0 YOUR 0 = EARLIER YOU SAID YOUR 3)
...)

Le mot clé "MY" va alors servir à insérer des phrases dans la mémoire. Lorsque ce mot clé est choisi comme le plus important par le mécanisme de classement à la fin de la lecture de l'entrée, l'une des transformations de la structure de mémoire est choisie aléatoirement. Une copie de l'entrée est alors transformée est sauvegardée dans une pile. Le reste de processus continue comme on l'a déjà décrit. Si jamais une entrée future n'admet aucun mot clé, une réponse est dépilée de la pile de mémoire et elle est envoyée à l'utilisateur. Ce mécanisme ajoute beaucoup à l'effet que ELIZA pouvait produire sur ces utilisateur.

Discussion

Dans son livre *Computer Power and Human Reason : From Judgement to Calculation*, Weizenbaum nous raconte des anecdotes parlant des premiers utilisateurs de ELIZA, qui devenait parfois attachés émotionnellement au programme. Ou l'exemple de sa secrétaire, qui a apparemment demandé à Weizenbaum de sortir de la pièce pour qu'elle puisse avoir une vraie conversation avec ELIZA.

> I had not realized that extremely short exposures to a relatively simple computer program could induce powerful delusional thinking in quite normal people.

On pourrait alors considérer ELIZA comme l'un des premiers programmes capables de tenter le test de Turing, et aussi capable de le passer sous certaines conditions. D'après une étude réalisée à UC San Diego qui demandait à ses volontaires de converser par message soit avec un programme, soit avec un vrai humain (les volontaires) ne savait pas quel était le cas, ELIZA était plus performante que GPT 3.5. Ceci est assez incroyable, considérant la simplicité et légèreté de ELIZA d'un point de vue informatique. Weizenbaum a réussi à créer un programme qui ne sait rien sur le monde réel, et qui arrive quand-même à tenir une conversation avec un humain en restant relativement convaincant. Tout ça en 1966 !

Il était intéressant de comparer ELIZA au programme de l'expérience de Georgetown-IBM. On voit que seulement en 12 ans, la complexité et les capacités des programmes ont augmenté énormément. Je m'en suis rendu compte aussi en écrivant ce rapport. Il m'a suffit de comparer le temps que j'ai mis à expliquer le fonctionnement du programme Georgetown-IBM, avec le temps que j'en ai mis pour ELIZA. Je trouve cette comparaison juste, car je crois que le niveau de détail que j'ai atteint dans les deux cas est à peu près le même.

ALPAC report

ALPAC (Automatic Language Processing Advisory Committee) était une commission de 7 scientifiques, dont le dirigeant John R. Pierce (un

ingénieur américain), 2 linguists, 2 chercheurs en traduction automatique (TA en abrégé), un psychologue et un chercheur en IA, crée par le gouvernement américain en 1964. Son but était d'évaluer le progrès de la recherche en linguistique informatique et plus spécifiquement en traduction automatique. Son rapport publié en 1966 est devenu célèbre pour sa critique de la TA comme domaine de recherche, ainsi que pour l'explicitation du manque de résultats utiles après une dizaine d'années de recherche.

Le rapport se concentre sur les applications pratique de la TA, et comme on se trouve aux États-Unis en 1966, ces applications consistent exclusivement à traduire des textes (surtout scientifiques) du russe vers l'anglais. Le rapport répète que l'utilisation des systèmes de TA tels qu'ils sont actuellement n'est pas envisageable. Les textes traduits automatiquement ne sont pas très compréhensibles et ils ont donc besoin d'être revu et corrigé par un humain. Ceci contredit la premise de la traduction automatique, qui justement est censée être *automatique*. D'après le rapport, il serait plus économique et efficace de garder la tâche de traduction chez les traducteurs humains, et de rediriger les financement et la recherche hors la traduction automatique.

Les effets de ce rapport ont été sévère pour le domaine. Il a des parties du rapport qui sont contestable. On peut aussi voir en le lisant qu'il essaye d'accentuer "l'échec" de la TA. Par exemple, le rapport fait une comparaison entre le programme de l'expérience Georgetown-IBM (dont on a déjà parlé) et les systèmes de TA actuels, en essayant de montrer que les résultats de ce premier étaient plus impressionnants que de ce dernier. Je pense que nous allons tous être d'accord qu'une telle comparaison est ridicule. Comparer un "programme-dictionnaire" capable de traduire 60 phrases spécifiquement choisies avec des systèmes qui traduisent des textes quelconques n'est pas très correcte. La rapport gonfle aussi les financements qui ont été attribués aux recherches en TA. 20 mil. de dollars d'après le rapport, mais en réalité il s'agissait plutôt de 12 ou 13 mil. de dollars.

On peut dire que le rapport n'a vu la traduction automatique que comme un outil des motivations stratégiques et politiques de l'époque, qui était déjà censé fonctionner, et qui en n'était pas capable. Il est vrai que les résultats de la TA étaient décevants, et qu'une application dans la vraie vie n'étais pas possible. Malheureusement, le rapport n'a pas reconnu le potentiel de la TA, et sa publication a beaucoup endommager le développement du domaine.

Approche statistique

Approche neuronale

Chapitre 3

Approche neuronale

Chapitre 4

Word2Vec

Chapitre 5

Deep Learning pour le TALN

Chapitre 6

Conclusion

Bibliographie

[1] Keith D. Foote. *A Brief History of Natural Language Processing*, juillet 2023.

[2] Université Masaryk, Brno, République Tchèque. *Centrum zpracování přirozeného jazyka (Centre de traitement du langage naturel)*, 2001-2023.

[3] Shamala Gallagher, Anna Raffert, Amy Wu, Sophomore College 2004 : The Intellectual Excitement of Computer Science, Professor Eric Roberts. *Natural Language Processing*, 2004.

[4] Les contributeurs de Wikipedia. *Natural language processing*, 2023.