Introduction aux langages réguliers, aux automates à états finis et aux expressions régulières, et leurs applications en machine learning

Pierre-François Gimenez

"Papers please", 15 décembre 2022

CentraleSupélec

Théorie des langages formels

Histoire

- L'étude des langages intéressent les linguistes depuis longtemps.
- Noam Chomsky a proposé le formalisme actuel en 1956
- Parseur LR (largement utilisé pour les langages informatiques) par Knuth en 1965
- Beaucoup de travaux dans la seconde moitié du XXe siècle
- Liens forts avec la théorie de la complexité (qui s'intéressent aux automates)

Objectifs

- Pouvoir décrire formellement des langages humains complexes (linguistique)
- Pouvoir analyser formellement les langages: savoir parser des phrases, en générer, avec des algos efficaces (ou démontrer qu'il n'existe aucun algorithme...)

Notions de bases

Définitions

- Un alphabet Σ est un ensemble fini de symboles.
- Ces symboles sont quelconques: il peut s'agir de lettres, de paquets réseaux, d'appels systèmes... la seule contrainte est qu'ils soient en **nombre fini**.
- Dans la suite, je vais utiliser les lettres `a`, `b`, `c`...
- Une phrase est une séquence (finie) de lettres. Par exemple "patate" est une phrase utilisant l'alphabet $\{a, e, p, t\}$. "mot" est un synonyme de "phrase".
- La phrase vide est notée ϵ
- Un langage est un ensemble (potentiellement infini) de phrases. Par exemple, $\{patate, raclette\}$ est un langage fini et $\{a, aa, aaa, aaaa, \ldots\}$ est un langage infini

Comment représenter et utiliser des langages infinis ?

Description mathématique

Intuition

On peut utiliser des notations mathématiques pour désigner des ensembles infinies. Par exemple :

- $\{a^n \mid n \geq 1\}$
- $\{w \mid$ chaque lettre apparaît autant de fois dans w$\}$
- $\{a^p \mid p$ premier$\}$
- $\{w \mid w$ est un programme C valide$\}$
- $\{w \mid w$ est une solution optimale au voyageur de commerce pour un certain graphe$\}$

Mais ceci ne nous aide pas vraiment pour connaître la forme des solutions, etc. De plus, il n'existe pas forcément de notation finies pour des langages infinies complexes !

Expression régulière

Définition

Une expression régulière est la description d'un langage à l'aide de trois opérateurs :

- la concaténation (noté $\cdot$, mais souvent implicite)
- la conjonction (noté $+$)
- l'étoile de Kleene (*) qui indique qu'un groupe peut être répété 0, 1 ou plusieurs fois

Remarque: en pratique, il y a des extensions avec plus d'opérateurs (négation, lookahead, etc.). En théorie des langages, on ne se base que sur ces trois opérations.

Exemple

- `abc` $=$ {abc}
- `ab(c+dd)` $=$ {abc, abdd}
- `a*(b+ccc)*` $=$ {abb, aaabcccb, b, ...}

CentraleSupélec

Notions de base

Grammaire

- Une grammaire est une description finie d'un langage (fini ou non), constituée d'un ensemble de règles de réécriture

Exemple

$\Sigma = \{$"Le", "La", "L'", "dort", "tue", "rédige", "doctorant", "article", "Moloch", "Toto"$\}$

- `<Phrase> → <Groupe nominal> <Verbe> | <Groupe nominal> <Verbe> <Groupe nominal>`
- `<Groupe nominal> → <Déterminant> <Nom commun> | <Nom propre>`
- `<Déterminant> → "Le" | "La" | "L'"`
- `<Verbe> → "dort" | "tue" | "rédige"`
- `<Nom commun> → "doctorant" | "article"`
- `<Nom propre> → "Moloch" | "Toto"`

Exemple

<Phrase> $\rightarrow$ *<Groupe nominal> <Verbe> <Groupe nominal>*
$\rightarrow$ *<Déterminant> <Nom commun> <Verbe> <Nom propre>*
$\rightarrow$ *"Le" "doctorant" "tue" "Moloch"*

Les phrases suivantes sont-elles générables par cette grammaire ?

- Le doctorant doctorant doctorant doctorant
- La doctorant dort l'article
- Moloch rédigea le doctorant

Remarques

- On commence toujours par le même symbole, appelé "axiome"
- Il y a deux types de symboles : les symboles terminaux (qui font partie de l'alphabet) et les symboles non-terminaux (symboles intermédiaires, entre <>)
- Les grammaires sont par nature des modèles génératifs

Grammaire régulière

Grammaire formelle

- On note toujours S l'axiome
- On note avec $a, b, c, \ldots$ les terminaux (symboles de l'alphabet)
- On note avec $A, B, C, \ldots$ les non-terminaux

Grammaire régulière

- Une grammaire est dite régulière si ses règles sont de la forme $A \to bC$ et $A \to \epsilon$
- À chaque étape de la dérivation, il y a au plus un non-terminal, et il est en dernière position
- Exemple : $S \to aA$, $S \to aB$, $A \to aB$, $B \to bS$, $S \to \epsilon$
- De quel langage s'agit-il ?

Automate : définition

Définition

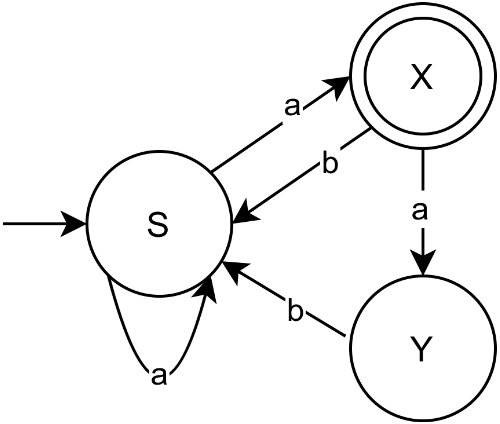
Un automate fini est décrit par:

- Un ensemble d'état
- Un état initial
- Un ensemble d'état finaux
- Une fonction de transition, qui est un ensemble de triplets *(état de départ, lettre, ensemble d'états d'arrivée)*

Automate déterministe et non-déterministe

- Un automate est dit déterministe si chaque transition arrive dans un seul état
- On peut transformer tout automate en automate déterministe en rajoutant des états (état dans l'automate déterministe = ensemble d'états dans l'automate non-déterministe)

Automate : exemple



Exemple

S est l'état initial, X l'état final. Cet automate est-il déterministe ?

Lesquels de ces mots sont reconnus par cet automate?

- aa
- aaaaaba
- ab
- aaabbaabaa

Langage régulier

Equivalence

On a présenté trois manières de décrire des langages infinies:

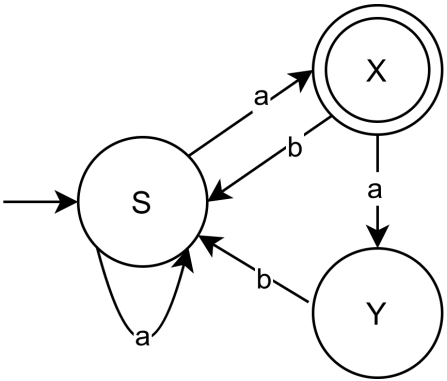
- les expressions régulières (faciles à lire/écrire pour un humain)
- les grammaires régulières (pratique pour générer des mots)
- les automates à états finis (pratique pour reconnaître des mots)

Ils décrivent la même classe de langage: on peut passer de l'un à l'autre selon les besoins.

Proprités

- Ces langages sont stables par concaténation, étoile de Kleene, union, intersection, complémentarité, image miroir...
- À peu près toutes les propriétés sont décidables

Equivalence automate $\Leftrightarrow$ grammaire régulière



On peut transformer un automate en grammaire régulière facilement

- Chaque état devient un non-terminal
- L'état initial est l'axiome de la grammaire
- Pour chaque transition (A, a, B) on ajoute la règle $A \to aB$
- Si A est un état final, on rajoute la règle $A \to \epsilon$

Inversement, on peut transformer une grammaire régulière en automate. L'équivalence avec les expressions régulières est légèrement plus complexe: cf. le théorème de Kleene

CentraleSupélec

Comment reconnaître un langage régulier?

Lemme d'itération / lemme de l'étoile

Soit L un langage régulier. Il existe un entier N tel que, pour tout mot $w \in L$ tel que $|w| \geq N$, il existe une factorisation $w = xyz$ (avec y non vide) tel que:

- $0 < |xy| \leq N$
- $xy^n z \in L$

Intuition: dans un automate à états finis, on change d'état à chaque nouvelle lettre. Avec un mot qui a plus de lettres qu'il y a d'états, alors forcément on va passer deux fois par le même état, ce qui veut dire qu'il y a une boucle dans l'automate. y correspond au mot qui réalise cette boucle. On peut faire autant de tours de boucle que l'on veut en ajoutant des y.

CentraleSupélec

Exercice !

Langage régulier ou pas ?

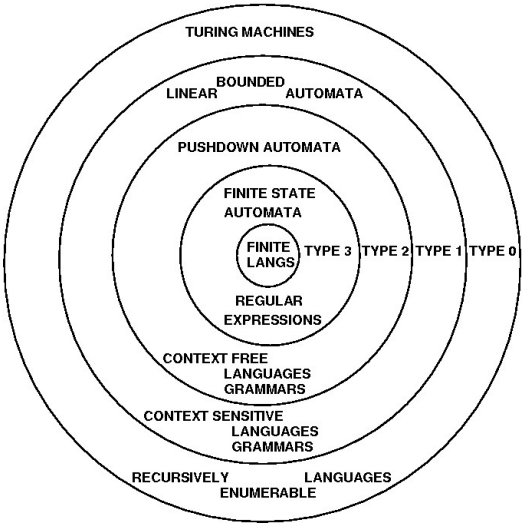
- $\{a^n b^n \mid n \geq 0\}$
- Un langage fini quelconque
- Langage des palindromes: $\{ww^r\}$
- L'ensemble des mots sur $\{a, b, c\}$ avec un nombre pair de a
- Langage des parenthèses équilibrées: $\{(), ()(), (()), ()()(()), \ldots\}$

Comment faire ?

- Si c'est régulier, proposer une expression régulière, une grammaire régulière ou un automate à états finis
- Si ce n'est pas régulier, utiliser le lemme de l'étoile

Attention: respecter le lemme de l'étoile n'implique pas être régulier !

Au-delà des langages réguliers



Et si on veut utiliser ces langages plus complexes?

- Il y a des langages plus complexes: algébriques, LR, LL, linéaire, déterministe, etc.
- Il n'y a pas vraiment d'équivalent aux expressions régulières pour ces langages-là
- Ils n'ont pas autant de propriétés que les langages réguliers

Lien avec le machine learning

Modèle en machine learning

- En machine learning, un modèle est une description succincte d'une réalité
- En apprentissage supervisé d'animaux, on représente de manière succincte des classes d'objets (des types d'animaux)
- En apprentissage binaire, on représente de manière succincte une ou deux classes d'objets
- Une classe d'objets est simplement un ensemble d'objets (par exemple, la classe "chien" parmi tous les animaux, etc.)
- Un langage est un ensemble d'éléments... un langage est un modèle !

$\rightarrow$ apprendre un langage revient à apprendre une classe! Utile pour la détection d'anomalies ou pour la génération...

Apprendre quoi ?

Grammaire, automate ou expression régulière ?

- On peut facilement passer de l'un à l'autre, mais la taille peut exploser (par exemple, un automate peut mener à une expression régulière exponentiellement plus complexe)
- L'état de l'art fait principalement de l'apprentissage d'automate
- En pratique, les automates appris ne sont pas lisibles par un humain

À partir de quelles données ?

- L'algorithme le plus répandu est L* et apprend à partir d'exemples positifs et négatifs en active learning (il pose des questions à un oracle)
- Des algorithmes supervisés qui ne sont pas en active learning existent
- Un algorithme one-class d'expression régulière existe [a]

[a] *Algorithms for learning regular expression from positive data*, Henning Fernau

Applications

Application à la génération de trafic réseau

- Une communication réseau est une succession de paquets entre A et B
- On peut modéliser un paquet par un tuple (direction, flags TCP) $\rightarrow$ c'est l'alphabet
- On apprendrait une expression régulière par protocole, vérifiable par un expert
- Exemple potentiel d'expression régulière apprise:
  `(F,Syn) (B,SynAck) (F, Ack) [ (F,Push) (B,no flag) ]* (F,Fin) (B,Ack) (B,Fin) (F, Ack)`

Avantages et limites des langages réguliers

Avantages

- Intersection possible avec d'autres langages régulier, par exemple "pas plus de trois Forward de suite", "au plus X forward et Y backward", etc.
- On peut générer des séquences de longueur quelconque

Limites

- Pas de probabilités d'apparition. Les chaînes de Markov sont une version probabiliste des automates finis
- Les expressions régulières ne peuvent pas exprimer tous les langages possibles...
- Les algos d'apprentissage ont tendance à trop généraliser

CentraleSupélec

Ce sur quoi on peut travailler

Ce qui manque à l'état de l'art

- L'expression régulière apprise explique tous les mots vus jusque là → y compris les cas exceptionnels / pas représentatif / etc.
- Il n'est pas possible d'apprendre l'expression régulière `(ab)*` → ce serait très utile pour modéliser des communications

⇒ on peut potentiellement faire mieux avec d'autres méthodes d'apprentissage, utilisant par exemple MDL (minimum description length)