Comment une IA découpe-t-elle un texte ?

Les tokens

Mise en situation. En 2024, de nombreux internautes se sont amusés à poser une question toute simple à des agents conversationnels : « Combien y a-t-il de lettres r dans le mot anglais *strawberry* (fraise) ? » Réponse fréquente : « deux »… alors qu'il y en a trois !
Comment une IA capable de rédiger une dissertation peut-elle se tromper sur une question aussi facile ? Parce qu'elle ne lit jamais les lettres : avant tout calcul, le texte est découpé en morceaux appelés **tokens**, par exemple `str` `aw` `berry`, et chaque morceau est remplacé par un **numéro**. Dans cette activité, vous allez découvrir comment on fabrique ces morceaux… et ce que cela change pour les réponses des IA.

1. Découper en mots ou en lettres ?

Définition

Un **token** est un morceau de texte : un mot entier, un morceau de mot, une lettre ou un signe de ponctuation. L'ensemble des tokens que connaît la machine s'appelle son **vocabulaire** ; chaque token y porte un **numéro**.
Le programme qui découpe un texte en tokens s'appelle un **tokeniseur**. Une IA ne reçoit jamais le texte lui-même, mais seulement la **suite des numéros** de ses tokens.

▶ À vous de jouer !

1. Dans la phrase « `Les chatons mangent des croquettes.` », combien y a-t-il de mots ? Combien y a-t-il de lettres ?
2. **Idée n°1 : un token par mot.** Supposons qu'une langue compte 8 000 verbes et que chacun possède environ 50 formes conjuguées (*je chante, nous chanterons, ils chantaient*…). Combien de formes faudrait-il numéroter, rien que pour les verbes ?
3. Avec un vocabulaire de mots, que se passe-t-il si la machine rencontre un mot qu'elle n'a jamais vu (*chocolatine*, un mot mal orthographié, un nom de personnage inventé…) ?
4. **Idée n°2 : un token par lettre.** Un texte de 1 000 mots contient environ 6 000 caractères (espaces compris). Découpé par un vrai tokeniseur, il donne environ 1 300 tokens. Combien de fois la suite à traiter serait-elle plus longue avec un token par caractère ? Arrondir au dixième.
5. Une IA ne peut lire qu'un nombre limité de tokens à la fois (sa « fenêtre de contexte »). Expliquer pourquoi ni les mots, ni les lettres ne sont un bon choix.

Coup de pouce

Pour la question 4, on divise le nombre de caractères par le nombre de tokens.

2. L'algorithme des fusions

Les tokeniseurs actuels utilisent une méthode appelée **fusion de paires** (en anglais *byte pair encoding*, BPE). On l'applique à un **corpus** : un très grand ensemble de textes. Ici, notre corpus est une simple liste de 17 mots :

chante chante chanter chanter chanter chantons chantons danse danser danser dansons dansons mange manger manger mangeons mangeons

On le résume dans un tableau, en indiquant combien de fois chaque mot apparaît dans le corpus :

Mot	chante	chanter	chantons	danse	danser	dansons	mange	manger	mangeons
Apparitions	2	3	2	1	2	2	1	2	2

Définition – l'algorithme des fusions

1. Au départ, chaque mot est découpé en lettres : le vocabulaire contient les 26 lettres, numérotées de 1 à 26.
2. On compte combien de fois chaque **paire de tokens voisins** apparaît dans le corpus, en tenant compte du nombre d'apparitions de chaque mot.
3. La paire la plus fréquente est **fusionnée** en un nouveau token, partout où elle apparaît. Ce token reçoit le numéro suivant (27, puis 28, etc.). En cas d'égalité, on choisit la paire rencontrée **en premier** en lisant le corpus dans l'ordre du tableau.
4. On recommence à l'étape 2 : c'est un **tour** de l'algorithme.

Exemple

La paire `c` `h` apparaît une fois dans *chante*, *chanter* et *chantons*. Comme ces mots apparaissent 2, 3 et 2 fois dans le corpus, la paire apparaît $2 + 3 + 2 = 7$ fois.

▶ À vous de jouer !

1. Compter de la même façon le nombre d'apparitions des paires `a` `n`, `e` `r` et `o` `n`. On admet qu'aucune autre paire n'apparaît plus de 11 fois : quelle paire est fusionnée au tour 1 ?
2. Écrire le découpage de chacun des 9 mots après le tour 1 (par exemple `c` `h` `an` `t` `e` pour *chante*).
3. Après le tour 1, quatre paires sont à égalité avec 7 apparitions : `c` `h`, `h` `an`, `an` `t` et `e` `r`. Laquelle est fusionnée au tour 2 ? Pourquoi ?
4. Effectuer les tours 3, 4 et 5 (au tour 5, la paire fusionnée apparaît 7 fois). Quels tokens a-t-on fabriqués ? Quel numéro porte chacun d'eux ?

Coup de pouce

Après une fusion, seules les paires qui touchent le nouveau token changent. Au tour 3, la paire `ch` `an` apparaît 7 fois et elle est rencontrée la première dans *chante*.

▶ À vous de jouer !

On poursuit l'algorithme jusqu'au tour 11. Les fusions des tours 6 à 11 sont, dans l'ordre : `on` (32), `ons` (33), `dan` (34), `dans` (35), `man` (36) et `mang` (37). Le corpus est alors découpé ainsi :

`chant` `e` `chant` `er` `chant` `ons` `dans` `e` `dans` `er` `dans` `ons` `mang` `e` `mang` `er` `mang` `e` `ons`

1. Que représentent les tokens `chant`, `dans` et `mang` ? Et les tokens `er` et `ons` ? La machine connaît-elle la grammaire ?
2. Pour découper un mot nouveau, on applique les 11 fusions **dans l'ordre où elles ont été apprises**. Découper les mots *mangerons*, *chanteur* et *rangeons*, puis écrire la suite de numéros correspondante (les lettres gardent leur rang dans l'alphabet : a = 1, b = 2, …, z = 26).
3. Pourquoi *rangeons* est-il découpé en plus de morceaux que *mangeons* ?

Remarque

Les vrais tokeniseurs appliquent cet algorithme à des milliards de mots et effectuent des dizaines de milliers de fusions : leur vocabulaire compte de quelques dizaines de milliers à quelques centaines de milliers de tokens. Les mots très fréquents (*le*, *chat*, *maison*…) deviennent un seul token, les mots rares sont découpés en plusieurs morceaux.

3. Ce que l'IA ne voit pas

▶ À vous de jouer !

1. Avec le vocabulaire de la partie 2, la machine reçoit le mot *chantons* sous la forme de la suite de numéros 30, 33. Peut-elle « voir » combien de lettres *n* contient ce mot ? Que doit-elle avoir appris pour répondre ? Expliquer alors l'erreur sur *strawberry*.

Certains tokeniseurs (c'est le cas de celui de GPT-4) découpent les nombres en paquets d'au plus 3 chiffres, en lisant de gauche à droite. Par exemple, 1234567 devient `123` `456` `7`.

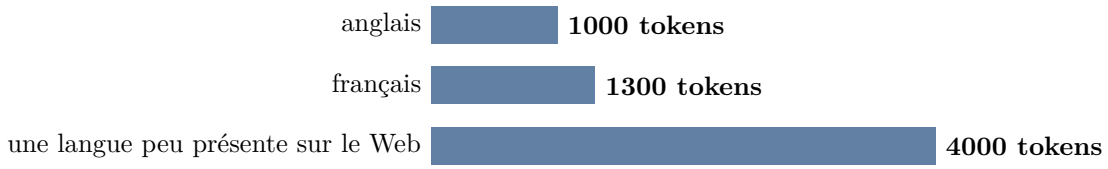
2. Découper de cette façon les nombres 2026 et 20260.
3. Dans 2026, que vaut réellement la partie représentée par le token `202` ? Et dans 20260 ? Le token `202` a-t-il toujours la même valeur ?
4. Découper 1234 et 567. Pour poser l'addition $1234 + 567$, on aligne les unités. Le chiffre des unités est-il, dans les deux nombres, un token à lui tout seul ? Pourquoi est-ce gênant pour calculer ?

Coup de pouce

On peut écrire $2026 = 202 \times 10 + 6$: le token `202` vaut donc 202×10.

4. Toutes les langues ne sont pas égales

On fait découper par un même tokeniseur une page de texte et ses traductions. On obtient les résultats suivants (ordres de grandeur pour l'activité, qui varient selon les tokeniseurs) :



▶ À vous de jouer !

1. De quel pourcentage le nombre de tokens augmente-t-il entre l'anglais et le français ?
2. Un service d'IA (fictif) facture 0,02 € pour 1 000 tokens. Calculer le prix du traitement de 50 pages dans chacune des trois langues.
3. La fenêtre de contexte d'une IA est de 8 000 tokens. Combien de pages **entières** peut-elle lire à la fois dans chaque langue ?
4. En vous appuyant sur l'algorithme des fusions, expliquer pourquoi l'anglais est découpé en moins de tokens que les autres langues. Pourquoi est-ce injuste ?

Coup de pouce

Pour la question 1, on calcule $\dfrac{\text{valeur d'arrivée} - \text{valeur de départ}}{\text{valeur de départ}}$.

Remarque

Des chercheurs ont mesuré ce phénomène en 2023 : selon les tokeniseurs, certaines langues demandent jusqu'à une dizaine de fois plus de tokens que l'anglais pour exprimer le même contenu.

5. Programmer en Python

Exemple

Pour compter combien de fois une paire de lettres apparaît dans un texte, on fait glisser une « fenêtre » de deux caractères : `texte[i:i+2]` est le morceau formé des caractères numéros `i` et `i+1`.

```
def compte_paire(texte, paire):
    n = 0
    for i in range(len(texte) - 1):
        if texte[i:i+2] == paire:
            n = n + 1
    return n

print(compte_paire("chantons", "on"))
```

▶ À vous de jouer !

1. Pour `texte = "chanter"`, que vaut `len(texte)` ? Quelles valeurs prend `i` ? Pourquoi la boucle s'arrête-t-elle à `len(texte) - 1` ?
2. Exécuter à la main `compte_paire("ananas", "an")` en complétant un tableau avec les colonnes `i`, `texte[i:i+2]` et `n`. Que renvoie la fonction ?
3. Que renvoie `compte_paire("chanter danser manger", "er")` ?
4. Écrire une instruction qui affiche le nombre de paires `"on"` dans le texte `"chantons dansons mangeons"`. Qu'affiche-t-elle ?

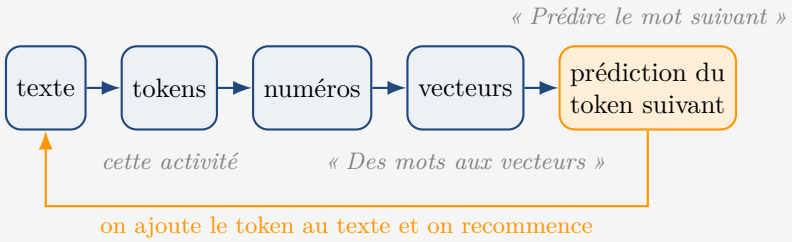
Coup de pouce

En Python, les caractères d'un texte sont numérotés à partir de 0 : dans `"chanter"`, `texte[0]` vaut `"c"` et `texte[0:2]` vaut `"ch"`.

6. Pour aller plus loin : la chaîne complète

Remarque

Les tokens sont la toute première étape du fonctionnement d'un modèle de langage, qui relie les activités de cette série :



★ Question étoilée

On reprend le corpus et les fusions de la partie 2.

1. Au départ, le vocabulaire contient les 26 lettres. Combien de tokens contient-il après 11 tours ? Que peut-on dire, en général, du nombre de tokens du vocabulaire après n tours ?
2. En comptant les 17 mots du corpus, celui-ci s'écrit avec 113 lettres. Combien de tokens faut-il pour l'écrire après le tour 1 ? Expliquer pourquoi chaque fusion fait diminuer ce nombre.
3. Après 11 tours, combien de tokens faut-il pour écrire le corpus ? Par combien a-t-on divisé la longueur du corpus ?
4. Un vrai tokeniseur part de 256 symboles de base (les 256 valeurs possibles d'un octet) et obtient un vocabulaire de 100 000 tokens. Combien de fusions a-t-il effectuées ?

À retenir

- Une IA ne lit ni des mots, ni des lettres, mais des **tokens** : des morceaux de texte, remplacés par des **numéros**.
- Le vocabulaire est fabriqué par l'**algorithme des fusions** : on fusionne, tour après tour, la paire de tokens voisins la plus fréquente du corpus. Les mots fréquents deviennent un seul token, les mots rares sont découpés en morceaux.
- Ce découpage a des conséquences : l'IA ne « voit » pas les lettres ni la valeur des chiffres, et les langues peu présentes dans le corpus sont découpées en plus de tokens, donc désavantagées.