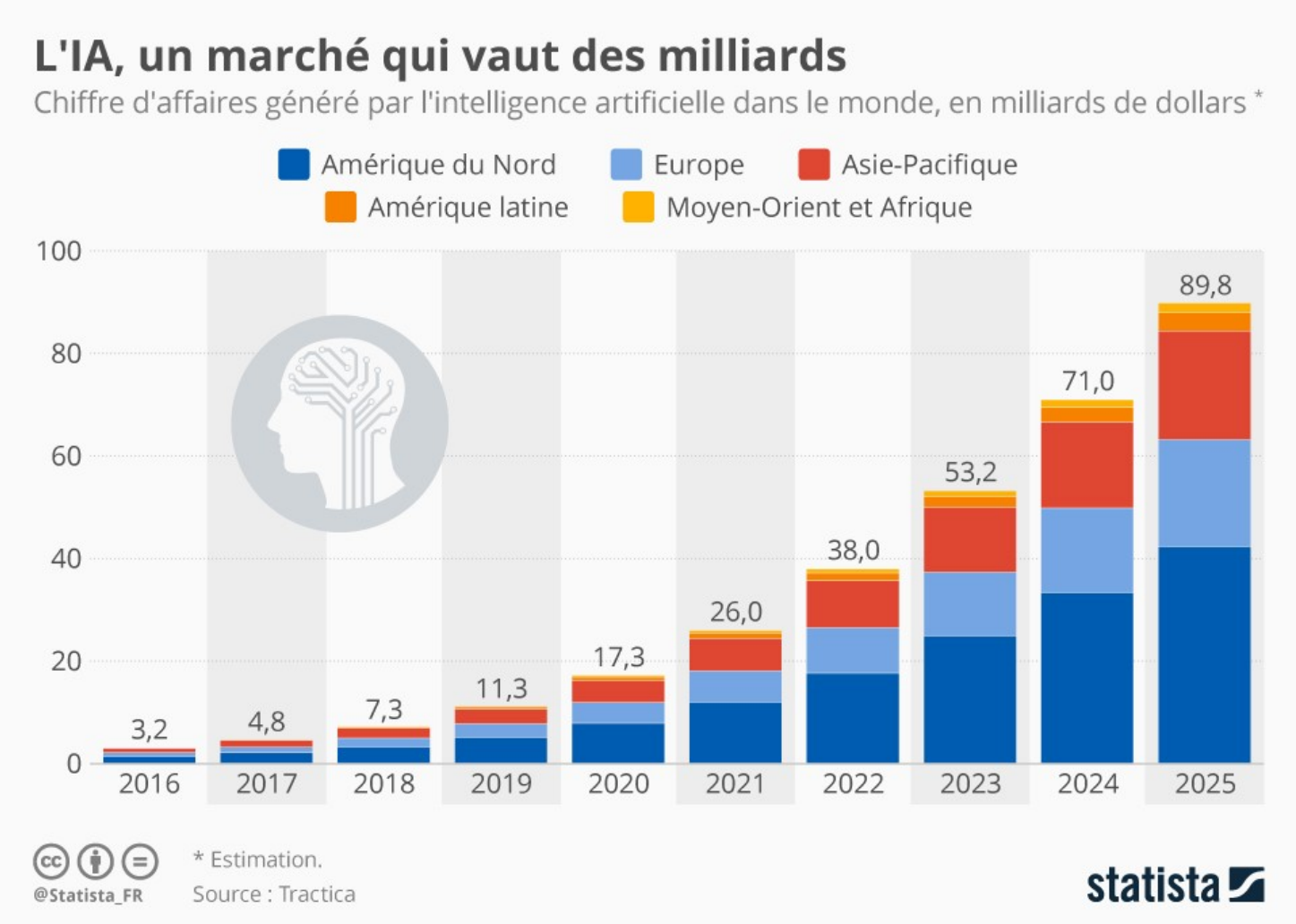
La révolution des Transformers dans les LLM

De la théorie mathématique des Transformers aux enjeux linguistiques dans un monde numérique multilingue

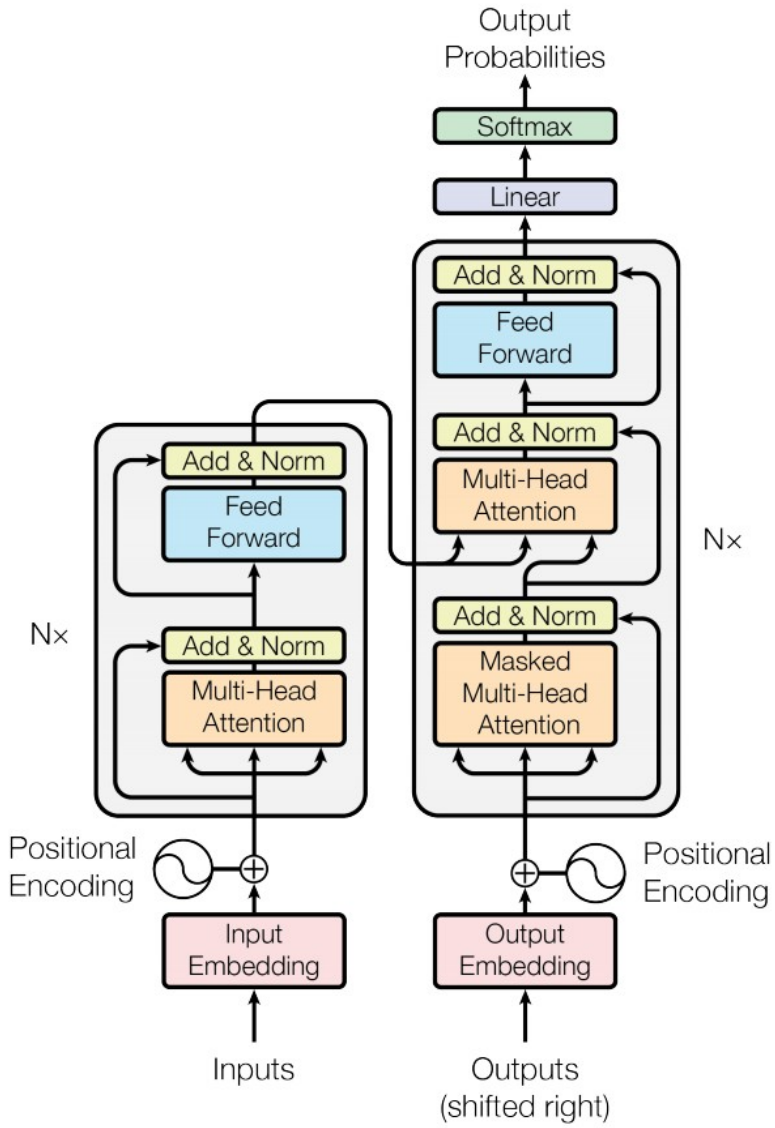
Contexte actuel et utilisation

- Secteur en grand essor
- Concurrence intellectuelle rude
- Utilisation diverse (imagerie médicale, prédiction monétaire, secteur privé etc...)



Qu’est ce qu’un Transformers

- Un **Transformer** est une architecture de réseau de neurones spécialisée dans le traitement des séquences de données



Invention et Inventeur

Le **Transformer** a été inventé par des chercheurs de Google Brain avec "*Attention Is All You Need*" publié en **juin 2017**.

Introduction du concept du **mécanisme d'attention**. Objectif de remplacer les réseaux récurrents (RNN) dans le traitement de séquences

Problématique

Les hypothèses des Transformers renforcent-elles des biais contre les langues complexes et peu utilisées ?

Plan

Analyse mathématique et technique du modèle Transformers

Impact des choix techniques sur le traitement du langage

Enjeux sociétaux et culturels des Transformers face aux inégalités linguistiques

Analyse mathématiques et Techniques

Principes de fonctionnement

Texte → Découpé en morceaux (**Tokenisation et vectorisation**)

Chaque mot → Comprend ses relations avec les autres (**Self-attention**)

Plusieurs analyses → Parallèles pour capter des aspects différents (**Multi-head attention**)

Ordre des mots → Pris en compte (**Position Encodings**)

Compréhension → Enrichie pour chaque mot (**Feed-forward layers**)

Résultat → Texte ou réponse final(e)

Tokenisation

Définition : Découpe le texte en unités (tokens) compréhensibles par la machine.

Pourquoi ? : Permet de traiter efficacement les mots, sous-mots ou caractères, surtout dans les langues avec vocabulaire large ou complexe.

Basée sur les mots : Chaque mot devient un token ("*Je mange*" → [Je, mange]).

Les Transformers utilisent le BPE

Sous-mots (BPE) : Découpe en fragments fréquents ("*mangeront*" → [man, ger, ont]).

Caractères : Chaque lettre est un token ("*manger*" → [m, a, n, g, e, r]).

Comment les mots deviennent des vecteurs ?

But :

- Transformer les tokens ou indices en **vecteurs numériques** continus que le modèle peut traiter.

Nature des résultats :

- Produit des vecteurs denses dans un espace multi-dimensionnel.
- Exemple :
  - Les → [0.12, 0.53, -0.34, 0.67]
  - chats → [0.91,

Lien avec le modèle :

- Les vecteurs sont optimisés pendant l'entraînement pour capturer des relations sémantiques.
- Proximité dans l'espace → Similitude

Exemples

On peut imaginer que le vecteur de "*boulangère*" est construit comme une combinaison de concepts tels que :

- **Profession** : "*boulanger*" (vecteur représentant le métier).
- **Genre** : "*féminin*" (vecteur représentant la féminisation).
- **Nature humaine** : "*personne*" (vecteur représentant un être humain).

Voir ça comme un espace où les mots proches sont proches de sens

Mot	Vecteur (exemple 4D)
boulanger	[0.8, 0.4, 0.7, 0.3]
féminin	[0.1, 0.2, -0.3, 0.4]
boulangère	[0.9, 0.6, 0.4, 0.7]

Ici, le vecteur de *"boulangère"* est obtenu en additionnant : boulangère=boulanger + féminin

L’attention multi-tête

1. **Attention** :
    1. Mécanisme permettant au modèle de **pondérer l'importance** de chaque mot en fonction des autres mots dans la phrase.

2. **Pourquoi multi-têtes ?**
    1. Permet de **capturer plusieurs aspects du contexte** simultanément.
    2. Exemple : comprendre "*Le chat mange*" sous différents angles : le sujet, le verbe, le contexte.

3. **Fonctionnement** :
    1. L'attention multi-tête génère **plusieurs représentations** du contexte, puis les combine pour une meilleure compréhension.

Séparation en "têtes" d'attention :

Chaque tête d'attention est responsable d'examiner un aspect particulier de la phrase. Par exemple, une tête pourrait se concentrer sur les relations syntaxiques (sujet/verbe), tandis qu'une autre pourrait se concentrer sur la signification sémantique des mots.

Calcul des pondérations :

Chaque tête calcule l'importance de chaque mot par rapport aux autres dans la séquence, en générant une **matrice d'attention**. Cette matrice indique la force de l'attention que chaque mot doit prêter à un autre.

Combinaison des informations :

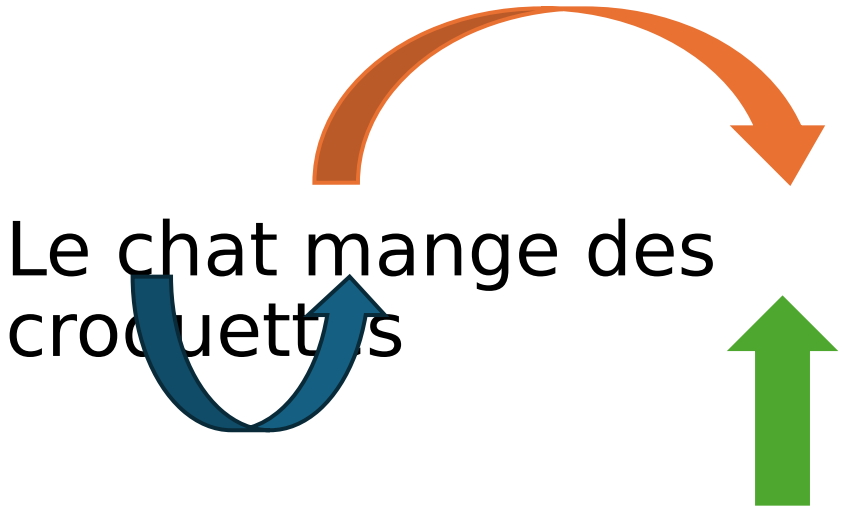
Après que chaque tête ait extrait ses informations, les résultats sont **concaténés** et passés à une couche suivante pour générer une représentation plus riche. C'est cette combinaison de perspectives qui permet une meilleure compréhension globale du texte.

Exemple

- Prenons la phrase :
  "Le chat mange des croquettes. »

Voici ce qui se passe dans un modèle avec attention multi-tête :

1. **Première tête** :
   Se concentre sur les relations **syntaxiques**. Elle pourrait attribuer une attention plus forte à la relation entre *chat*(sujet) et *mange* (verbe).
2. **Deuxième tête** :
   Se concentre sur les relations **sémantiques**. Elle pourrait attribuer une attention plus forte entre *mange* et *croquettes*, reconnaissant que ces deux mots forment une action spécifique.
3. **Troisième tête** :
   Peut se concentrer sur un autre aspect, par exemple sur la **structuration temporelle** ou l'importance d'une certaine information dans un contexte plus large (comme une attention sur "croquettes", relevant qu'elles sont l'objet de l'action).



Flexibilité, performants et meilleure gestion des structures complexes

Position Encodings

Le problème sans Position Encodings :

- Les Transformers n'ont pas de notion intrinsèque de **l'ordre des mots** dans une phrase.

Les Position Encodings :

- **Ajout d’une information sur la position des mots** dans la séquence.
- Permet au modèle de **comprendre le sens** en fonction de l'ordre des mots (ex. : sujet-verbe-objet).

Comment ça marche :

- Les Position Encodings sont ajoutés aux vecteurs des mots pour **incorporer des informations d’ordre** tout en maintenant leur signification sémantique.

Exemple

- *Sans Position Encoding* : Les mots "Le", "chat", "mange" ont des embeddings indépendants.
- Exemple : "Le chat mange" et "Mange le chat" seraient traités de manière identique.
- *Avec Position Encoding* : Chaque mot reçoit un encodage supplémentaire en fonction de sa position dans la séquence, ce qui permet de préserver l'ordre des mots.

Les Feed Forward Layers

Rôle des Feed Forward Layers :

Après que l'**attention multi-tête** combine les informations de contexte, les Feed Forward Layers affinent ces informations.

Chaque vecteur mot passe par une **transformation non linéaire indépendante** dans cette couche.

Fonctionnement :

Deux transformations :

- Une **projection linéaire** vers un espace de dimension plus grande (expansion).
- Une réduction de la dimension originale après application d'une **activation non linéaire** (ReLU).

Ces transformations permettent au modèle de capturer des relations **non linéaires complexes** entre les mots.

Indépendance par position :

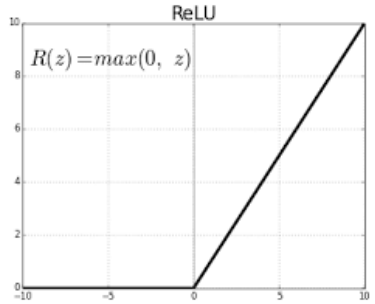
Chaque mot est traité **indépendamment des autres**, garantissant que le traitement est parallèle et rapide.

Exemple

Input : Vecteur pour "*chat*" → [0.3, 0.5, 0.2]

Feed Forward Layer :

- Étape 1 : Expansion linéaire(1) → [1.2, 0.8, 0.5, 1.0]
- Étape 2 : ReLU activé → [1.2, 0.8, 0.5, 1.0
- Étape 3 : Réduction linéaire → [0.7, 0.6, 0.3].



(1) :

Impact des choix techniques sur le traitement de langage

Biais induits par la structure de traitement

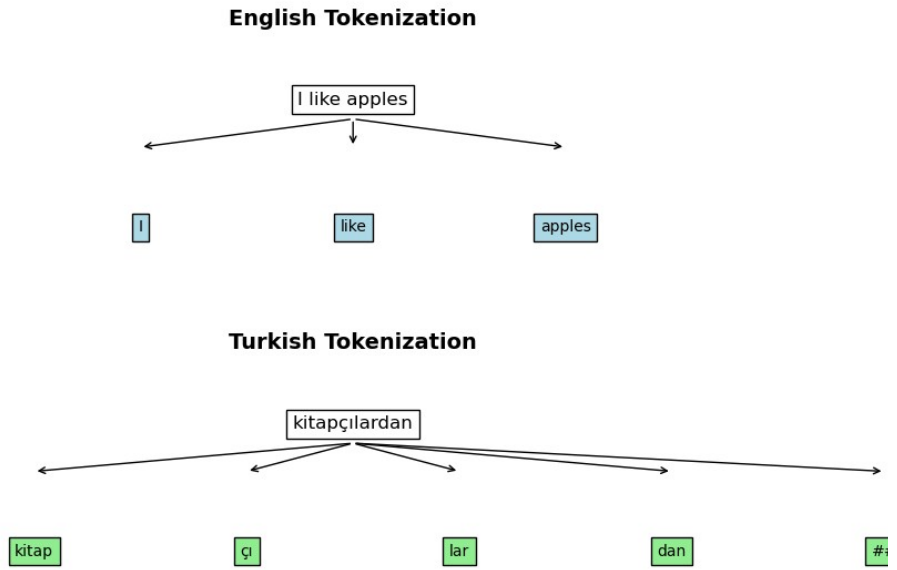
Créé par et pour des langue construite comme l'anglais

Les **choix techniques** des Transformers, comme la **tokenisation** et les **Position Encodings**, peuvent introduire des biais linguistiques.

Ces biais peuvent **favoriser certaines langues** et nuire à d'autres, particulièrement celles ayant des structures morphologiques complexes.

Comment la tokenisation favorise certaines langues

- Les langues avec une structure simple (anglais, espagnol) sont mieux segmentées.
- Les langues agglutinantes (turc) ou sans espaces (chinois) posent des défis.
- Impact : perte d’information sémantique pour certaines langues.



Limites des Position Encodings

Les Position Encodings ajoutent une notion d’ordre.

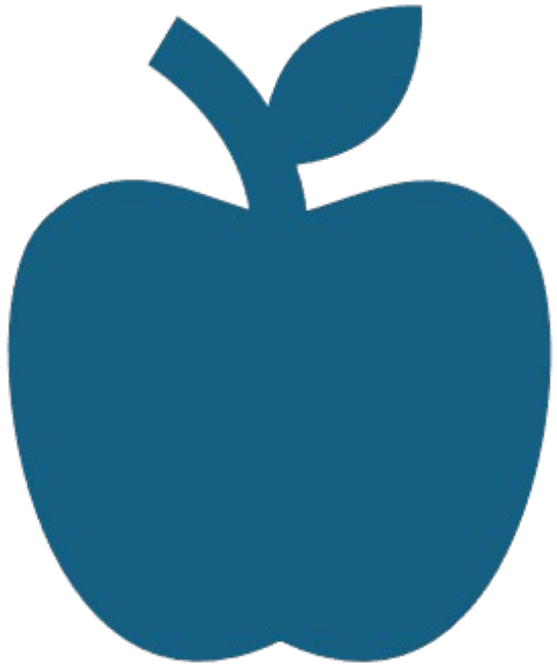
Problème : inefficaces pour des langues à structure flexible comme le japonais.

Exemple : "*Je mange une pomme*" → structure SVO (anglais) vs. SOV (japonais).

Exemple

- **Anglais : "I like apples"**

1. **Phrase initiale :** "I like apples".
2. **Tokens générés :** ["I", "like", "apples"].
3. **Encodage positionnel :**
    1. "I" reçoit une position 1.
    2. "like" reçoit une position 2.
    3. "apples" reçoit une position 3.
4. Chaque mot est traité comme une unité individuelle, ce qui est simple pour les modèles comme les Transformers.



Exemple

- **Japonais : " 私はりんごが好きです "**

1. **Phrase initiale :** " 私はりんごが好きです " (Watashi wa ringo ga suki desu, qui signifie "J'aime les pommes").
2. **Tokens générés :**
    1. Avec une tokenisation en caractère, cela devient : [" 私 ", " は ", " りん ", " ご ", " が ", " 好 ", " き ", " です "].
    2. Les caractères kanji (" 私 ", " 好 ") et hiragana (" は ", " りん ") sont traités comme unités distinctes.
3. **Encodage positionnel :**
    1. " 私 " (Watashi) reçoit une position 1.
    2. " は " (wa) reçoit une position 2.
    3. " りん " (rin) reçoit une position 3, et ainsi de suite.
4. La difficulté est que les modèles doivent comprendre à la fois les kanji (mots porteurs de sens) et les hiragana (particules grammaticales). Cette complexité n'existe pas en anglais.

Défis pour les langues à morphologie complexe

- Langues agglutinantes (hongrois, turc) : morphèmes multiples en un mot.
- Langues tonales (chinois, thaï) : le ton change le sens d’un mot.
- Solutions partiellement efficaces : tokenisation par caractères, encodages appris.

Langues sans séparation claire des mots

- Langues comme le chinois ou le thaï : pas d’espaces → ambiguïtés dans la segmentation.
- Solutions : segmenter par caractères ou créer des encodages contextuels.
- Limite : perte de relations sémantiques.

Les biais amplifiés par le déséquilibre des données

Les données d’entraînement sont majoritairement en anglais, espagnol, chinois.

Les langues minoritaires manquent de corpus riches et variés.

Conséquence : résultats médiocres pour ces langues, renforçant les inégalités.

Importance de la quantité de données d’entrainement

- Toutes les avancées majeures dans l’IA sont le fruit de la création d’une base de données étiquetée (Antonio Casili).
- Le chinois n’est pas censé bien fonctionner, mais compensation par la quantité de données.
- LE facteur clef

Performances des Transformers sur les langues minoritaires

- Écart de précision entre langues majoritaires et minoritaires.
- Exemple : traduction ou analyse syntaxique peu fiables pour des langues comme le wolof ou le maori.
- Impact sur l’accès aux outils linguistiques.

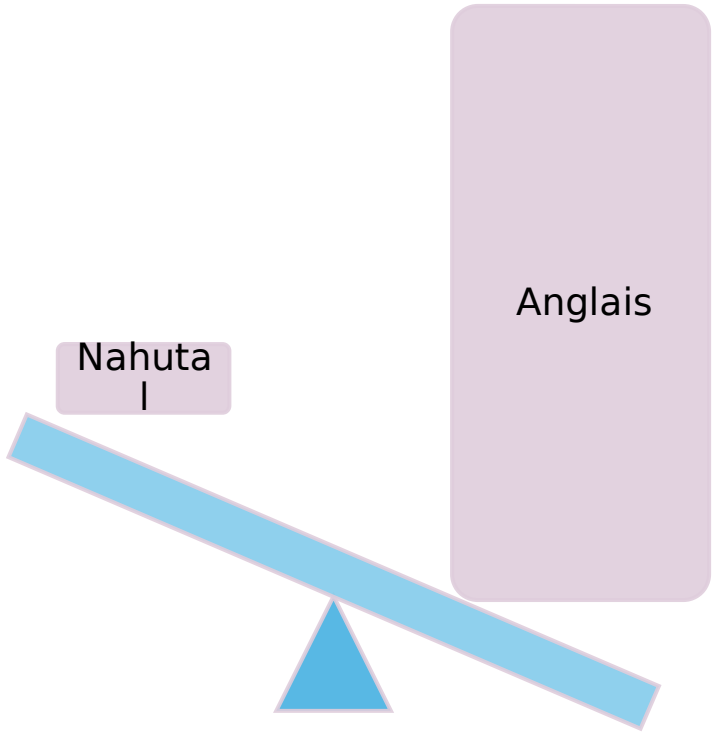
Enjeux sociétaux et Culturels des Transformers face aux inégalités linguistiques

Hégémonie des langues dominantes

- 70% des données d’entraînement des modèles multilingues sont en anglais.
- Les performances des modèles sont souvent inférieures pour les langues minoritaires.
- Exacerbation des disparités dans les contextes académiques, économiques et sociaux.

Menace pour la diversité linguistique et culturelle

- Déclin de l'utilisation des langues moins représentées en ligne.
- Perte progressive de traditions et cultures transmises via ces langues.
- Exemple : Domination de l'anglais dans les forums, les publications scientifiques et les réseaux sociaux.



Une fracture numérique accrue

- Les outils comme Google Translate ou ChatGPT fonctionnent mieux en anglais qu’en zoulou ou basque.
- Impact sur l'éducation : accès limité à des ressources automatisées dans les langues locales.
- Freins à l’inclusion numérique des communautés marginalisées.

Des voix étouffées

- Les langues dominantes gagnent une visibilité mondiale disproportionnée.
- Les locuteurs des langues minoritaires peinent à s'exprimer ou à trouver une reconnaissance en ligne.
- Risques de biais amplifiés dans des domaines comme l’intelligence artificielle générative (exemple : texte généré en anglais avec des nuances culturelles non universelles).

Qui doit agir ?

- Intégration de la diversité linguistique comme critère de développement technologique.
- Pression des institutions internationales pour financer les ressources linguistiques pour les langues sous-représentées.
- Obligation éthique des grandes entreprises d'IA (OpenAI, Google, Meta) d’inclure des solutions multilingues inclusives.

Augmentation des corpus multilingues

- Création et utilisation de bases de données pour des langues sous-représentées.
- Crowdsourcing des locuteurs natifs pour enrichir les datasets.
- Développement de projets open-source collaboratifs (ex. FLORES-200 de Facebook).

Optimisation des modèles existants

- Développement de modèles régionaux ou spécifiques par langue.
- Encodages appris qui adaptent la position et la sémantique pour chaque langue.
- Préférences pour les approches « low-resource » adaptées aux petites bases de données.

- Exemple du projet Masakhane pour les langues africaines.

Lutter contre l’injustice linguistique

- Coopération entre gouvernements, universités et ONG pour promouvoir l'équité linguistique.
- Éducation sur l'importance de la diversité linguistique dans le développement de l’IA.
- Exemples de projets communautaires locaux (ex. développement d’applications dans des langues autochtones).

Conclusion

- Un avenir linguistique équitable ?
  - Les Transformers doivent s’adapter pour refléter la richesse linguistique mondiale.
  - Importance de l’engagement continu des chercheurs et ingénieurs pour réduire les biais.

- Comment intégrer plus d’inclusion linguistique dans l’IA tout en respectant les contraintes techniques et économiques ?