



RAPPLE 2.0

Marie FELLER, Quentin HOUVENAGHEL, Flavian
MASDEBRIEU, Hélène Cucchi, Oussama NAJI

Sommaire

1. Contexte & problématisation
2. Démarche & structure
3. Résultats & démonstration
4. Conclusion & perspectives

Contexte & problématisation

2018 → Rapple “1.0”

2025 → team Rapple 2.0



L'équipe



Flavian



Hélène



Marie



Oussama



Quentin

Contexte & problématisation

“Créer une version améliorée de RAPPLE : RAPPLE 2.0”



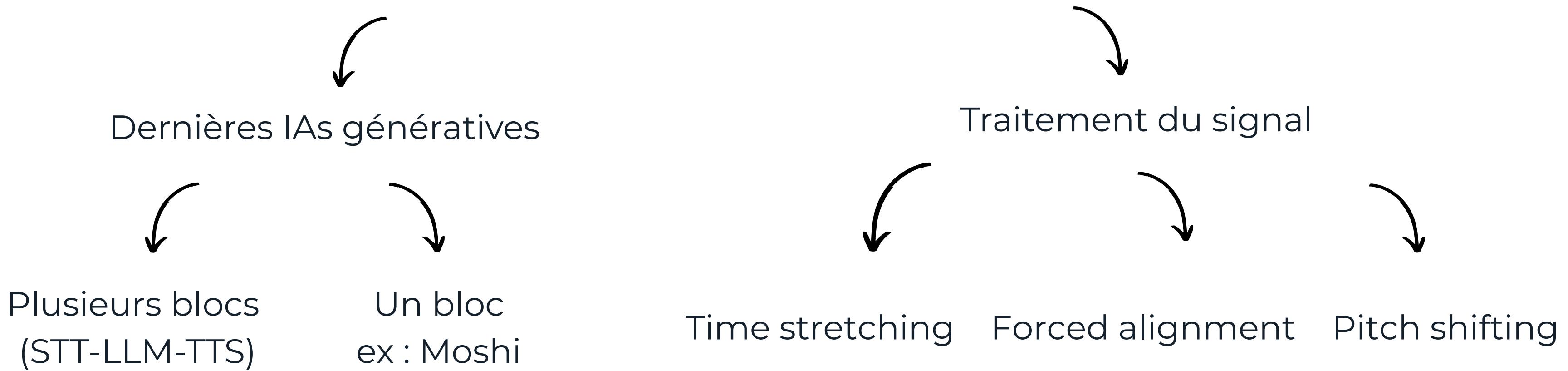
Dernières IAs génératives



Traitement du signal

Contexte & problématisation

“Créer une version améliorer de RAPPLE : RAPPLE 2.0”



Contexte & problématisation

Objectifs :



Qualité de la réponse



Temps réel



Flow

Contexte & problématisation

Objectifs :



Qualité de la réponse



Temps réel

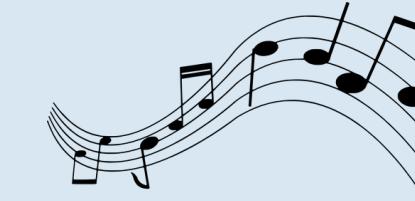


Flow

Le flow ?

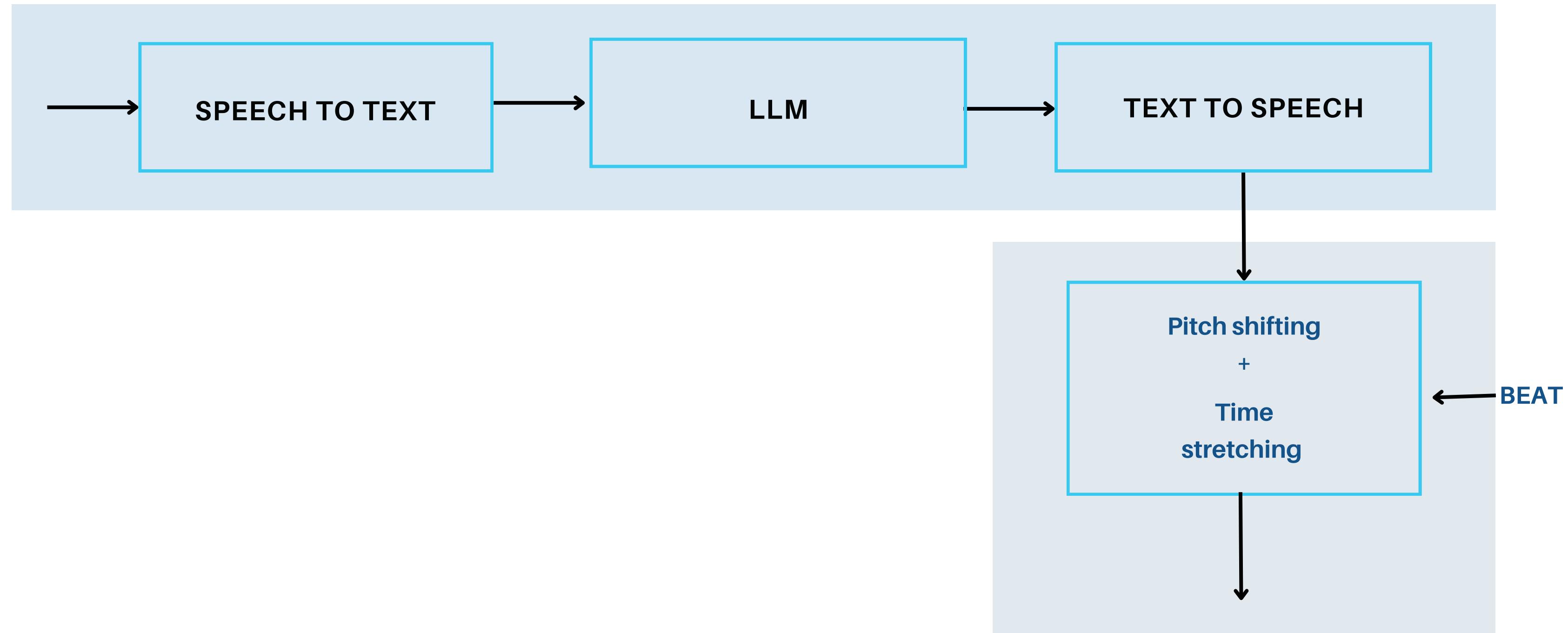


Rythme & variations

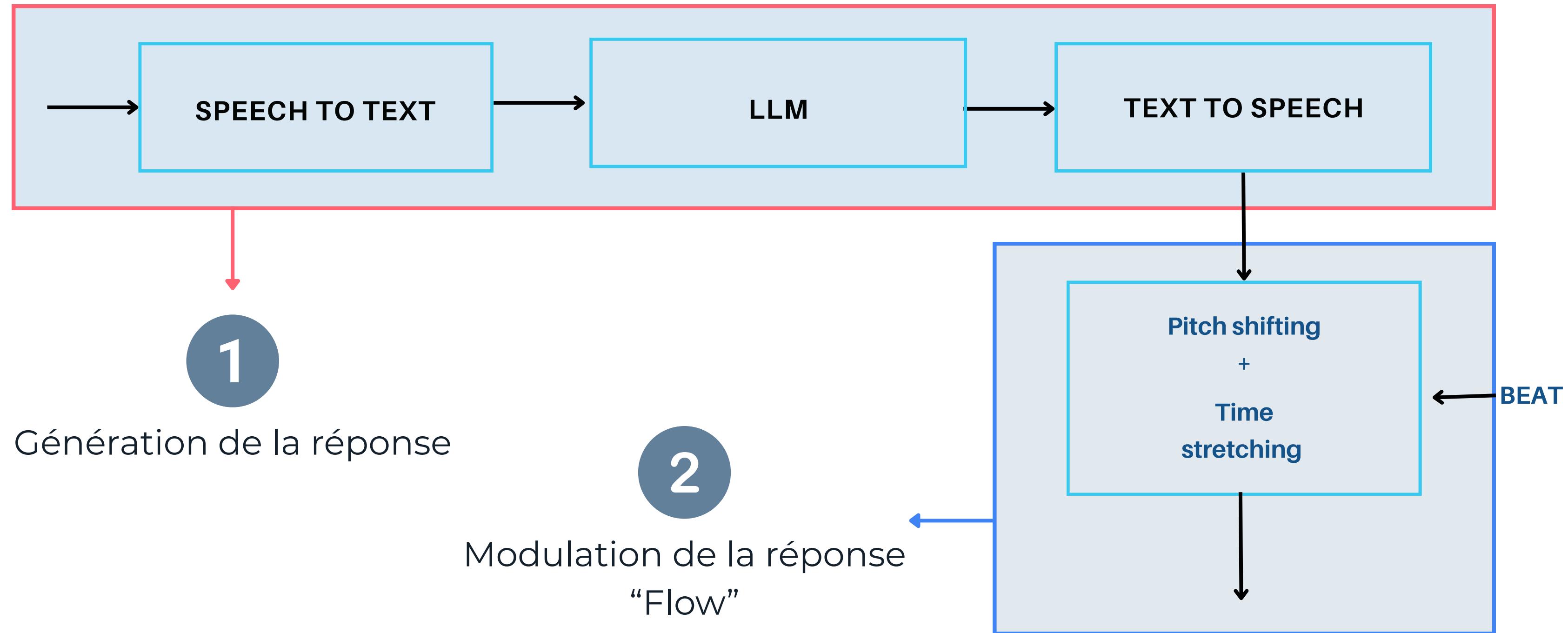


Mélodie

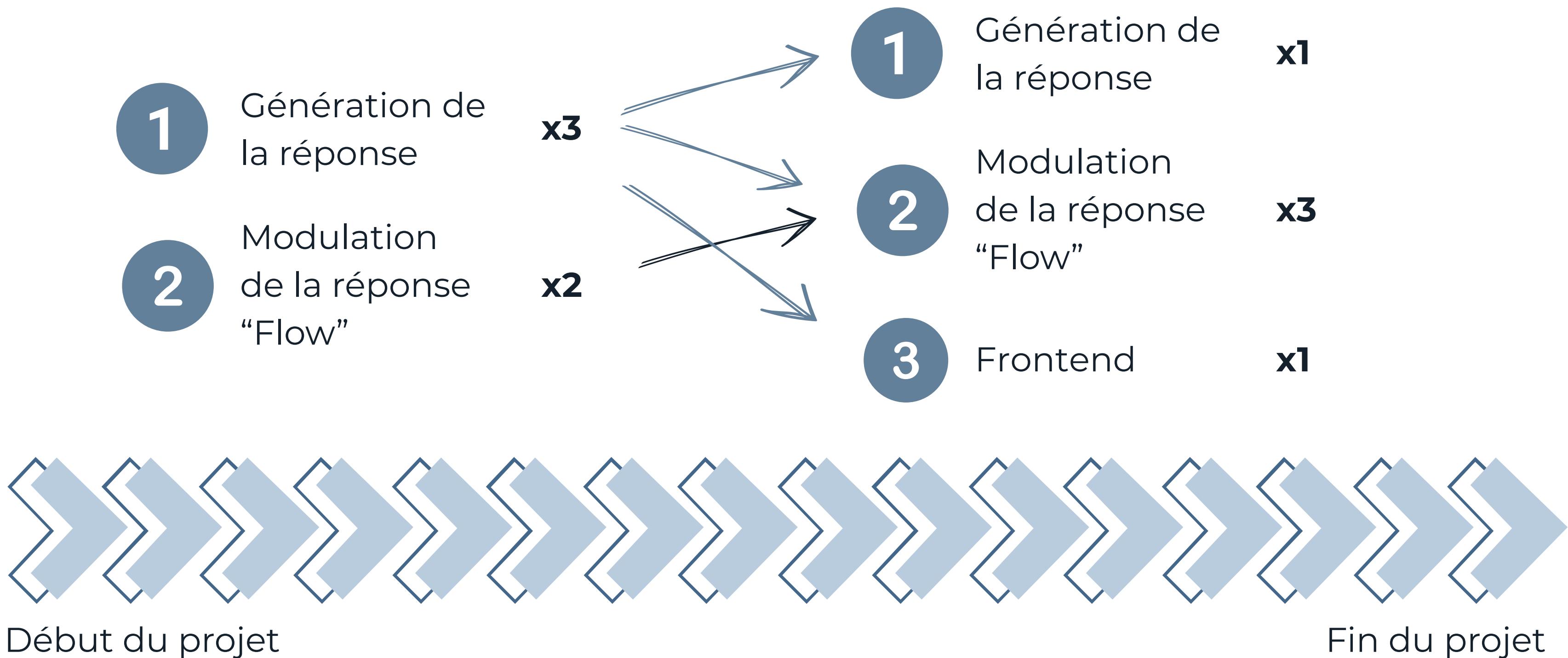
Démarche & structure



Démarche & structure



Démarche & structure



Démarche & structure

1

Génération de la réponse



Spectrogramme
Log-Mel



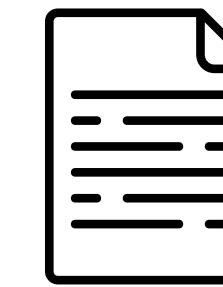
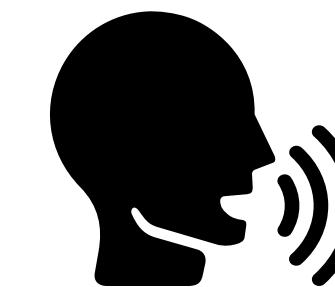
Encoder transformer

Vecteurs



Decoder transformer

Texte



Démarche & structure

1

Génération de
la réponse



Autoregressive text generation

GPT-2: a unidirectional autoregressive model

$$P(x) = \prod_{t=1}^T P(x_t | x_1, x_2, \dots, x_{t-1})$$

Where:

- x : the full sequence of tokens (e.g., a rap verse)
- T : total number of tokens
- x_t : the token at position t
- $P(x_t | x_1, \dots, x_{t-1})$: probability of generating token x_t given all previous tokens

Example prompt:

"You're in a brutal rap battle. Your opponent just said:
'You rhyme like a snail with a stutter.'
Respond with 2–4 bars that destroy them.

1. Encode the prompt into tokens

2. Generate the next token using:

$$x_{n+1} \sim P(x_{n+1} | x_1, \dots, x_n)$$

$$x_{n+2} \sim P(x_{n+2} | x_1, \dots, x_{n+1})$$

$$x_{n+3} \sim P(x_{n+3} | x_1, \dots, x_{n+2})$$

3. Add this token to the prompt

4. Repeat until reaching max_length or an end-of-sequence token



Sampling token-by-token

Encode the prompt

(x_1, x_2, \dots, x_n)

Text	Token ID
You're	837
in	287
a	257
brutal	6763
rap	1274
battle	4813

Compute

$$P(x_{n+1}|x_1, \dots, x_n) = \text{Softmax}(Wh_n, \tau)$$

Token ID	Score (Wh_n)	Probabilité
837	3.5	0.35
287	3.1	0.29
257	2.6	0.18
6763	2.3	0.10
1274	1.9	0.06
4813	1.8	0.05

$$P_i = \frac{e^{z_i/\tau}}{\sum_{j=1}^V e^{z_j/\tau}}, z_i = Wh_i \text{ (score brut)}$$

- **W: matrice de projection de sortie dans GPT-2** de taille $V \times d$, où V = taille du vocabulaire (ex : 50 000)
- **h_t**: vecteur de contexte de dimension **d** (dimension des embeddings)
- **tau**: température (1.75 → **Plate**) → Aucun token n'est **ultra-dominant et créativité**
- Sampling **top-p (0.9)** → **Évite les aberrations (eg. 'uh', ou des quotes)**
- **Décision:** Le prompt $x_{\{n+1\}}$ qu'on choisit correspond à la probabilité maximale (avec **do_sample=False**) mais on peut faire mieux...

Décision en pratique : top-p sampling

- On trie tous les tokens par probabilité décroissante.

$$P = \{(x_1, p_1), (x_2, p_2), \dots, (x_V, p_V)\} \quad \text{avec} \quad \sum_i p_i = 1$$

- On calcule la somme cumulée de leurs probabilités.

$$p_{(1)} \geq p_{(2)} \geq \dots \geq p_{(V)}$$

- On garde juste assez de tokens pour que leur somme atteigne au moins p .

$$T_p = \{x_{(1)}, x_{(2)}, \dots, x_{(k)}\} \quad \text{tel que} \quad \sum_{i=1}^k p_{(i)} \geq p$$

- On échantillonne aléatoirement parmi ces tokens sélectionnés.

$$x_t \sim \text{CATEGORIELLE}(T_p)$$

Token ID	Score (Wh_n)	Probabilité
837	3.5	0.35
287	3.1	0.29
257	2.6	0.18
6763	2.3	0.10
1274	1.9	0.06
4813	1.8	0.05

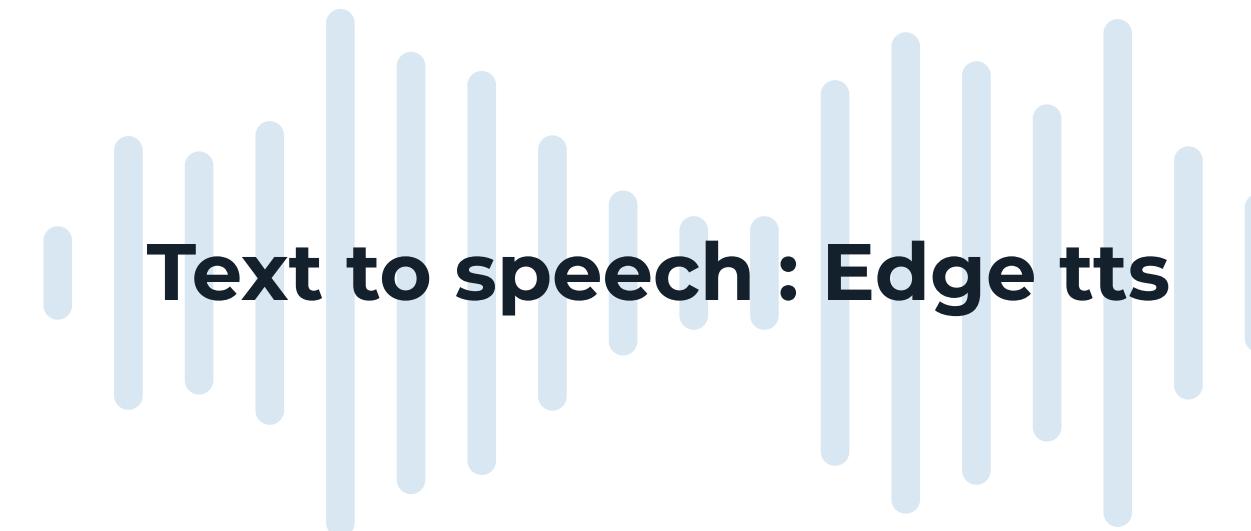
➡ Ici, dès qu'on atteint 0.90, on arrête : les 5 premiers tokens sont gardés, les autres sont ignorés.

👉 Le modèle échantillonne aléatoirement parmi ces 5, favorisant variété tout en restant dans les choix les plus crédibles.

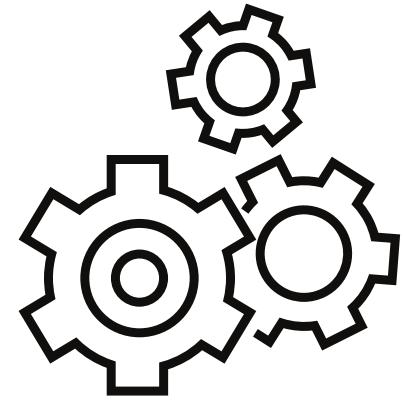
Démarche & structure

1

Génération de
la réponse



Point Technique - Requête API



“Application Programming Interface”

Type *REST (HTTP)*

Le serveur, comme Flask ou Gentle, est un programme qui tourne en continu et attend des requêtes.

L'API est l'ensemble des points d'entrée (les routes/endpoints comme /align, /upload, etc.) qu'il expose au monde extérieur.

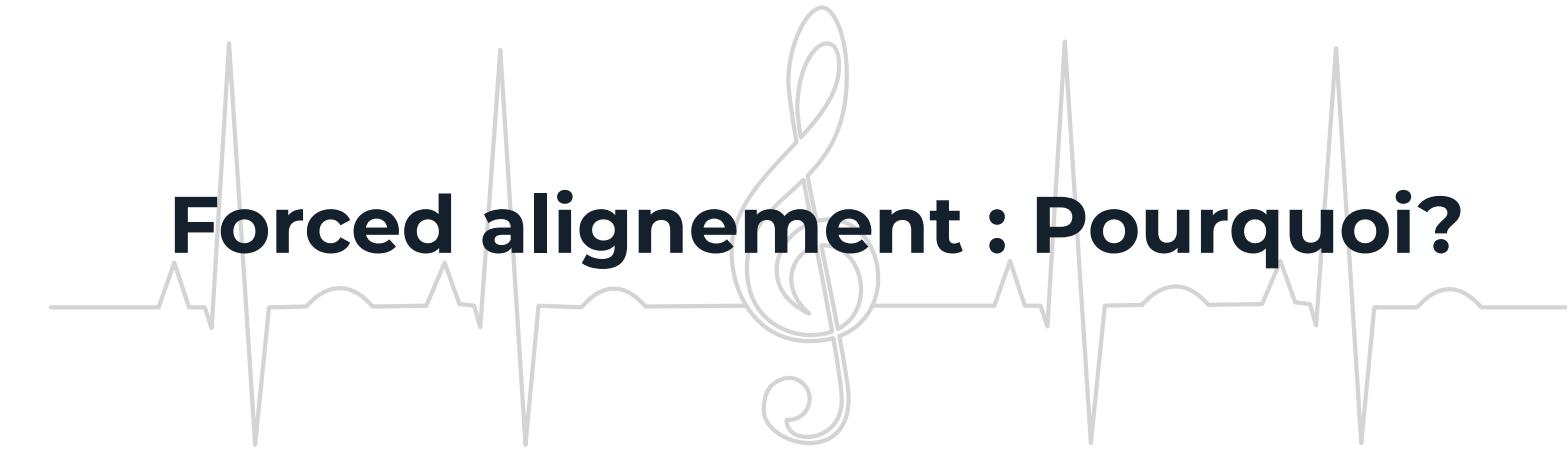
Requête API {

- Méthode :** GET, PUT, POST
- Adresse :** <http://api.example.com/transcribe>
- Corps :** Json, fichier audio...

Démarche & structure

2

Modulation
de la réponse
“Flow”



Obtenir précisément à quel moment est prononcé quel mot dans l'audio.

En entrée :

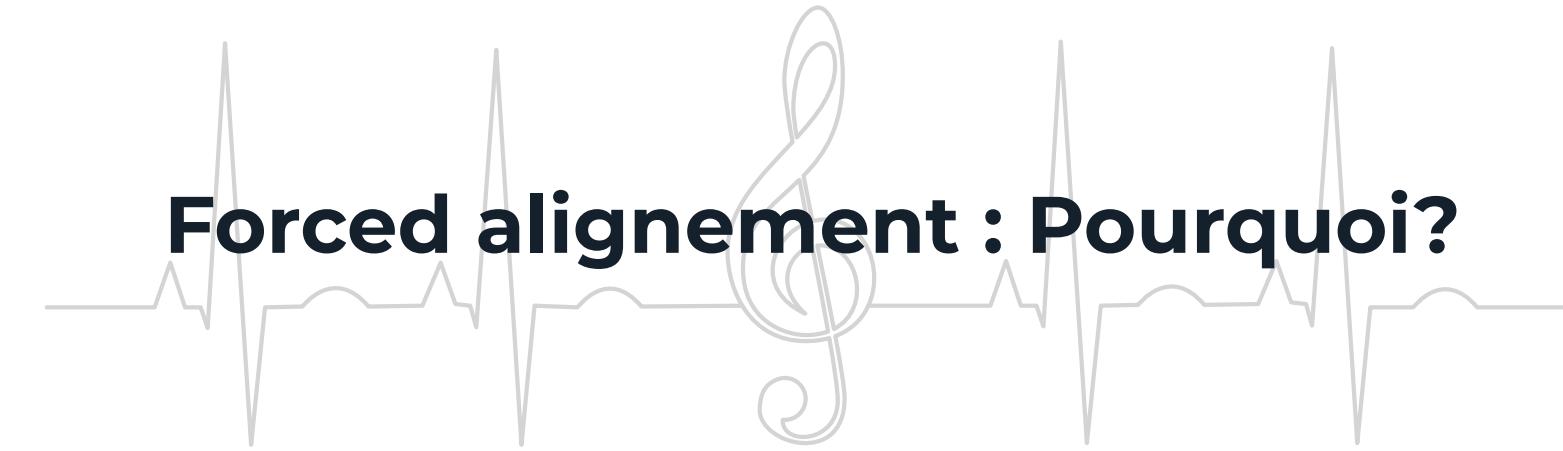
- **L'audio du rap IA**
- **Le transcript de l'audio ci dessus**

En sortie : {'alignedWord': 'vacant', 'case': 'success', 'end': 18.299999, 'endOffset': 244, 'phones': [{'duration': 0.08, 'phone': 'v_B'}, {'duration': 0.06, 'phone': 'ey_I'}, {'duration': 0.06, 'phone': 'k_I'}, {"duration": 0.06, "phone": "ah_I"}, {"duration": 0.04, "phone": "n_I"}, {"duration": 0.03, "phone": "t_E"}], 'start': 17.969999, 'startOffset': 238, 'word': 'vacant'}

Démarche & structure

2

Modulation
de la réponse
“Flow”



Obtenir précisément à quel moment est prononcé quel mot dans l'audio.

En entrée :

- **L'audio du rap IA**
- **Le transcript de l'audio ci dessus**

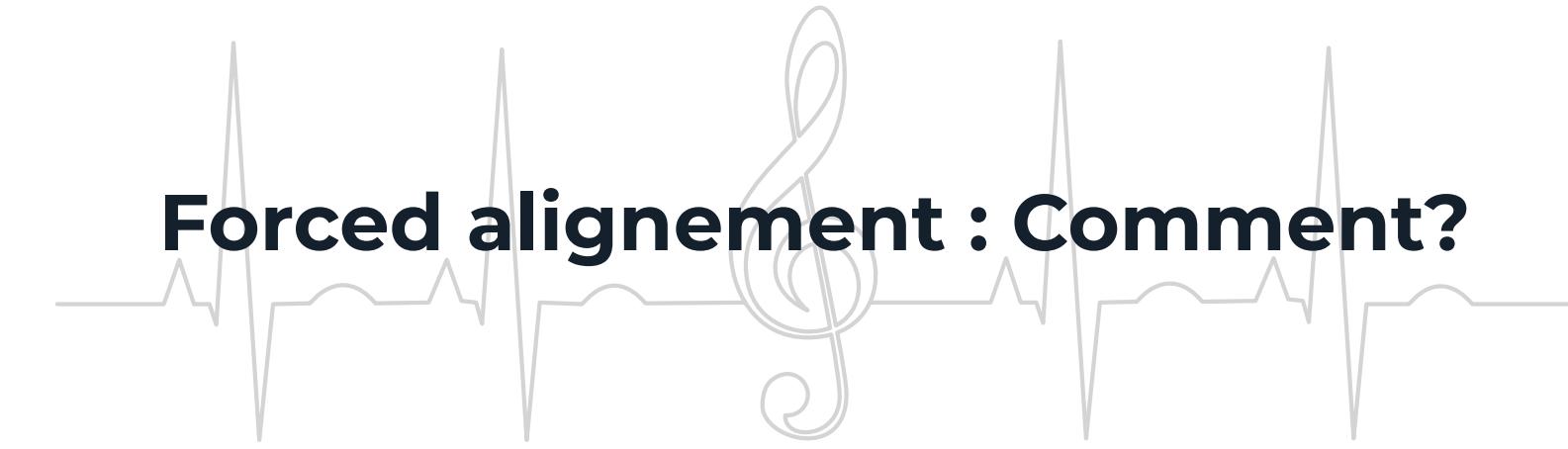
En sortie (ce qui est gardé):

```
{"end": 18.299999, "start": 17.969999, "word": "vacant"}
```

Démarche & structure

2

Modulation
de la réponse
“Flow”



Fonctionnement de Gentle:

- 1 - Découpage du texte en mots puis en phonèmes (CMU Dictionary)
- 2 - Envoie d'une partie de 25ms d'audio a Kaldi (Facebook wav2letter)
- 3 - Kaldi prend le phonème possible qui y ressemble le plus
- 4 - Algorithme Viterbi afin d'obtenir le chemin le plus probable

Démarche & structure

2

Modulation
de la réponse
“Flow”



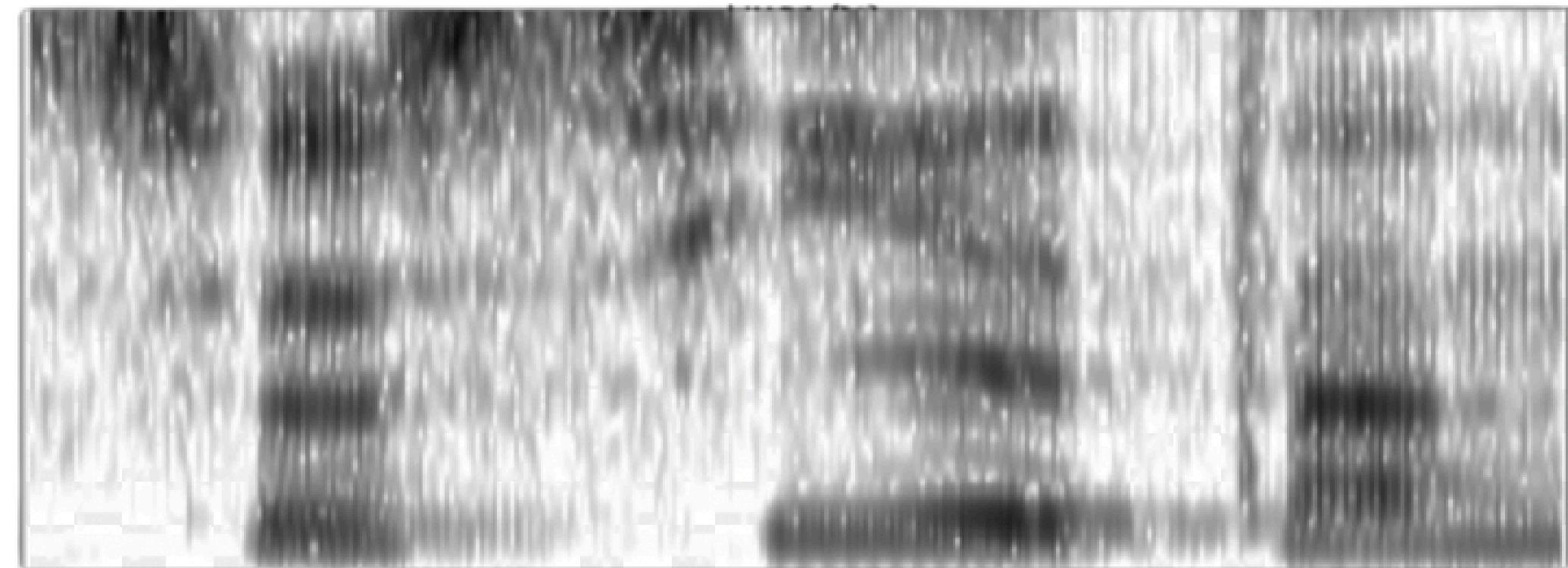
Permet de modifier la longueur d'un fichier audio sans modifier le “pitch”

Objectifs: Ajouter un flow (un rythme) pour faire “chanter” la voix parlée

Démarche & structure

Comment fonctionne le time stretching?

- L'audio est transformé en **spectrogramme**
- Les tranches temporelles sont **décalées**, tout en respectant les périodes de phase, grâce au **Phase Vocoder**
- Reconstruction de la piste audio avec une transformée inverse



Spectrogramme

Démarche & structure

2

Modulation
de la réponse
“Flow”



Global Stretch : Librosa; Pourquoi?

Dans une optique rythmique, on souhaite :

- Que chaque phrase **commence et tienne dans un multiple du temps d'une mesure**
- Qu'il y ait une **pause d'une mesure** entre chaque phrase
- Que chaque phrase occupe le **même nombre de mesures**

Démarche & structure

2

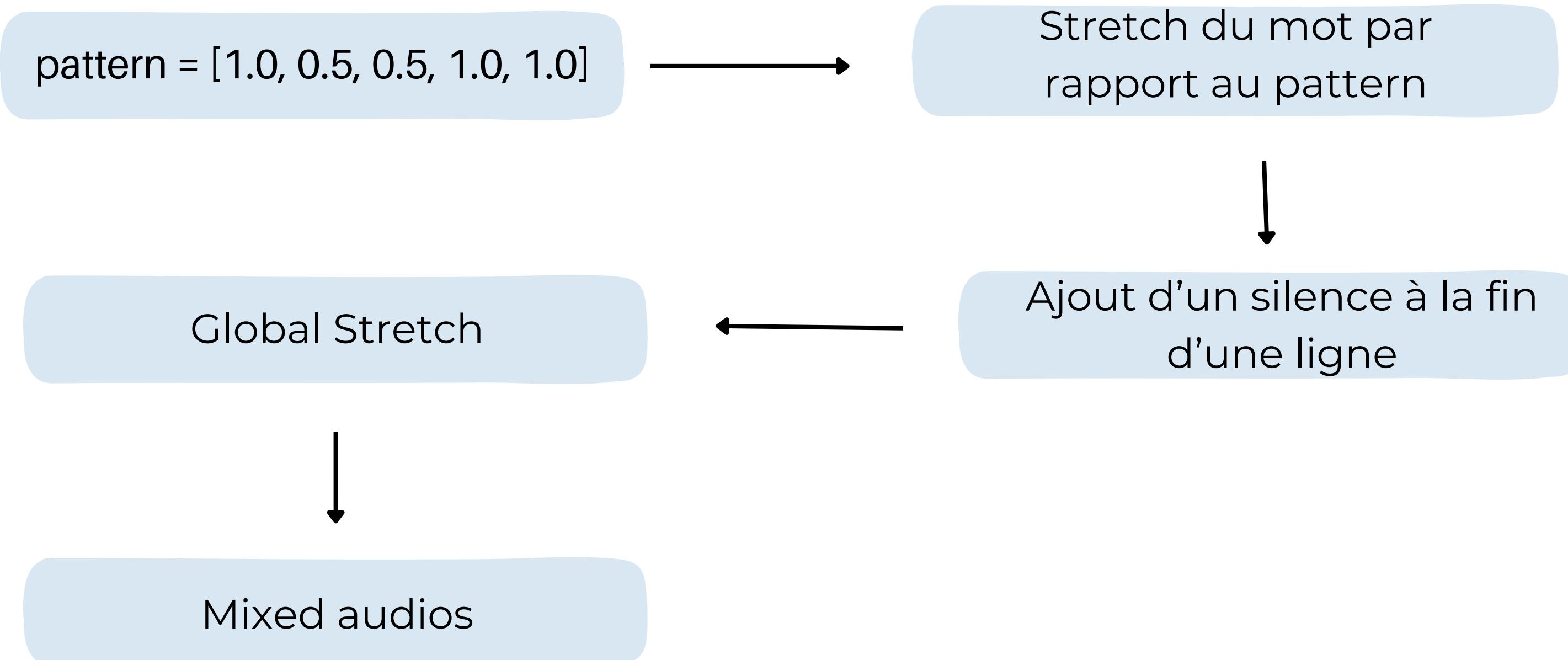
Modulation
de la réponse
“Flow”



Le fonctionnement :

- 1 - **Obtenir les timestamps** des temps de silence correspondant aux phrases
- 2 - Calculer les durées des phrases et **déterminer pour chacune la durée visée**
- 3 - Choisir au choix le min, le max ou la médiane des durées
- 4 - **Calculer le ratio** (end-start)/durée_visée **et appliquer** pour chaque phrase
- 5 - **Rajouter un silence** de durée une mesure

Démarche & structure



Démarche & structure

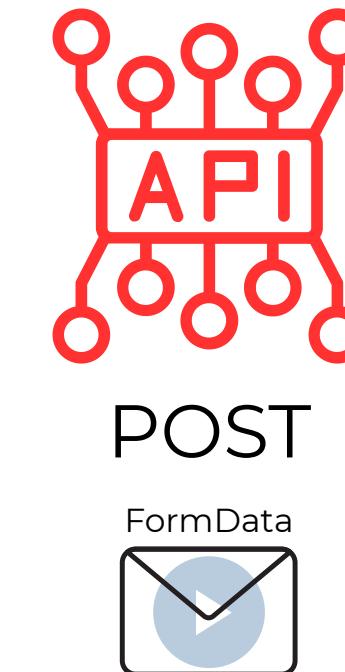
3

Frontend

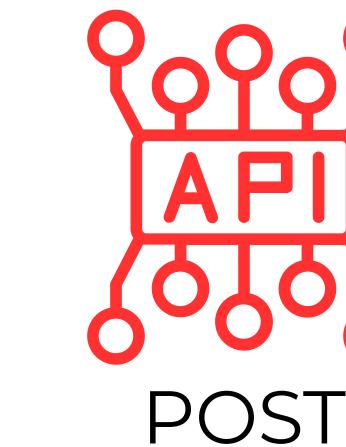


BLOB
BINARY
LARG
OBJECT

webn



AI
backend

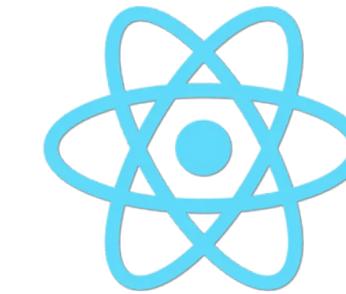


BLOB
BINARY
LARG
OBJECT

webn



Frontend : React



Résultats & démonstration

🎤 **Quand tu es prêt pour rapper,
clique sur le bouton !**

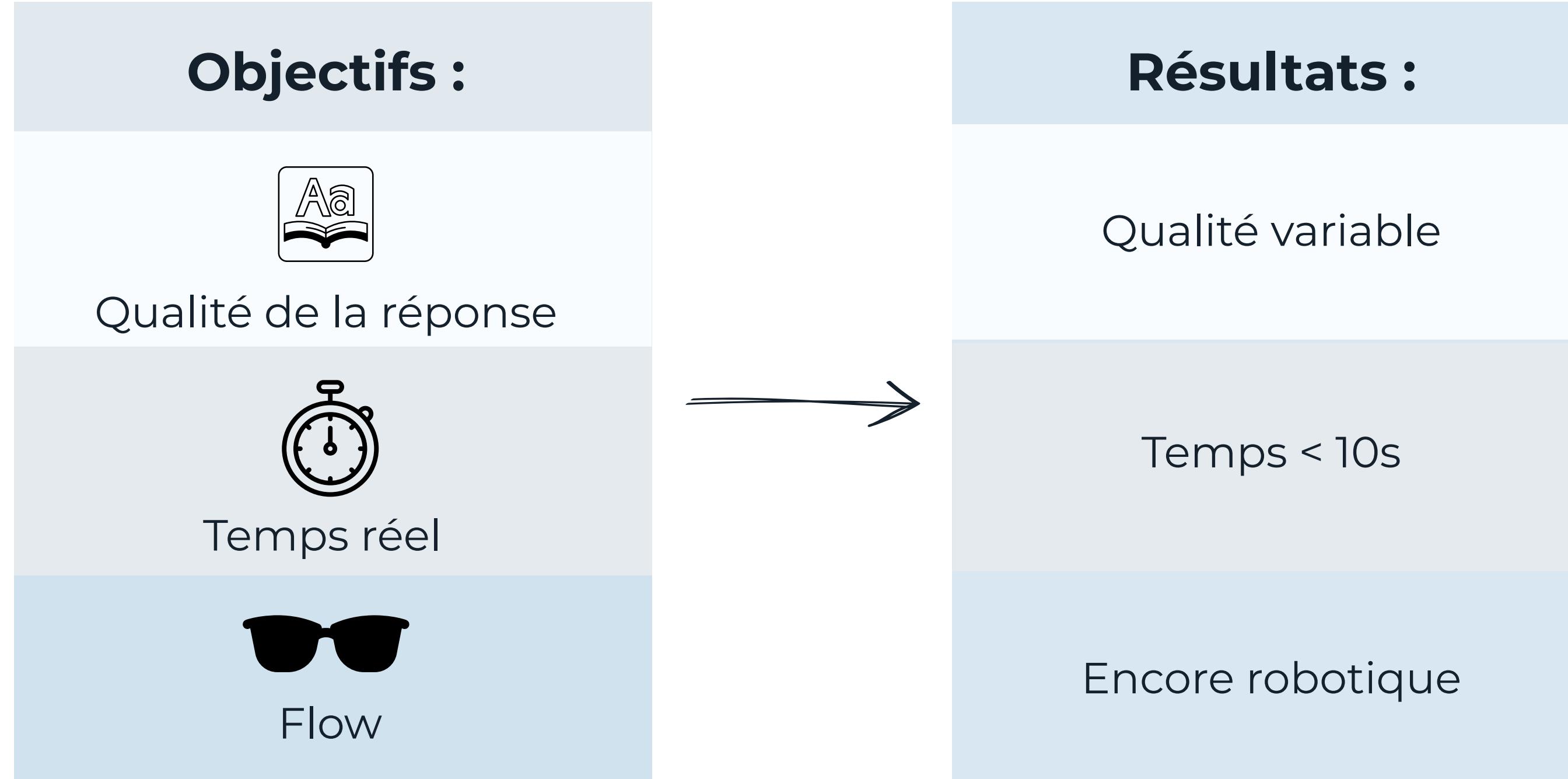
00:00:07:70

Je suis prêt !

▶ 0:00

Télécharge ta performance

Résultats & démonstration



CONCLUSION

Piste d'amélioration : Amélioration du temps de réponse
Écoute permanente
LLM spécialisé

CONCLUSION

Hard skills :

Maîtrise d'outils de développement
-Front
-Back
-Requêtes
-IA

Soft skills :

Répartition et gestion des responsabilités
Capacité d'adaptation
Leadership
Proactivité et prise d'initiative



MERCI !

ANNEXES

Spectrogramme Log-Mel

