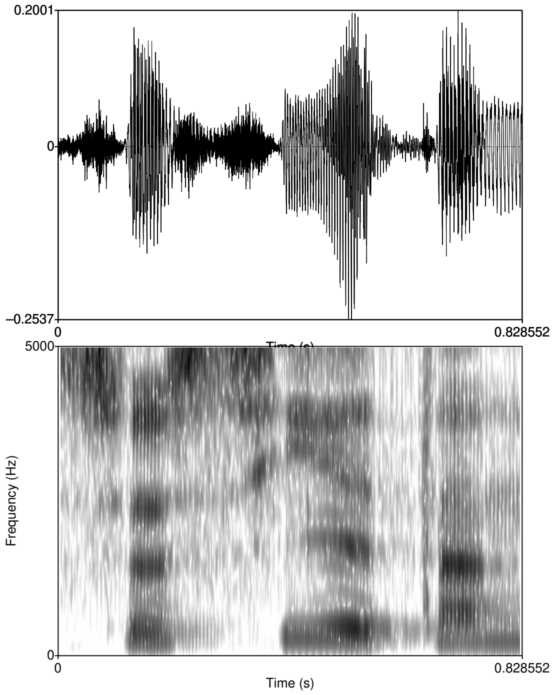
Travaux pratiques d'expérimentation humaine : montage d'une manip de restauration phonémique

Christophe Pallier[1]

1 La robustesse du signal de parole

La parole est transmise par un signal acoustique — c'est à dire une onde longitudinale de pression (cf. figure 1) — qui fait vibrer le tympan, mouvement transmis à la cochlée par les os de l'oreille moyenne.

FIG. 1 – Oscillogramme et spectrogramme d'une onde acoustique correspondant à un morceau de phrase



L'intelligibilité de la parole est remaquablement robuste aux distortions du signal acoustique.

[1]http ://www.pallier.org

Par exemple, le signal de parole utilise essentiellement les fréquences comprises entre 100 et 5000 Hz, mais on peut supprimer une large partie de ces fréquences et le signal demeure compréhensible [Fletcher, 1929]. On peut écréter le signal, voire le discrétiser complétement (remplacant les valeurs positives par +1 et les valeurs négatives par -1) [Licklider, 1946, 1950], et il demeure compréhensible.

Quand on interrompt ou qu'on inverse le signal dans des tranches de plusieurs dizaines de millisecondes, la compréhension peut demeurer très bonne Miller and Licklider [1950], Saberi and Perrot [1999].

Cela suggère que le signal de parole est très *redondant*, et aussi, sans doute, que le cerveau « va au delà » des données, c'est à dire interprète un signal incomplet.

L'un des arguments les plus souvent cités en ce sens est le phénomène de *restauration phonémique* mis en évidence par Warren [1970] . Il s'agit d'une illusion dans laquelle on entend des sons de parole qui ont en fait été « enlevés » du signal et remplacés par du bruit blanc. Typiquement, les personnes décrivent entendre une phrase intacte avec un bruit superposé.

Le but ce TP est de construire des stimuli et une expérience simple de restauration phonémique similaire à celle décrite dans Samuel [1981].

Pour créer les stimuli, on va utiliser le logiciel de manipulation de sons Praat développé par Paul Boersma and David Weenink à l'université d'Amsterdam.

L'expérience proprement dite sera programmée dans le language Python avec la librairie pygame.

2 À vous de jouer : manipulation de sons avec Praat

1. Téléchargez les fichiers sons (au format .wav) :
   – http ://www.pallier.org/misc/tp2/modif.wav
   – http ://www.pallier.org/misc/tp2/modif_nonoise.wav
   – http ://www.pallier.org/misc/tp2/intact.wav
2. Ecoutez plusieurs fois le premier, puis les deux autres, successivement. Vos impressions ?
3. Télécharger le logiciel de phonétique « Praat » à partir du site http ://www.praat.org.
4. Ouvrir et examiner les fichiers sons précédents (Menu `Read from file`, puis bouton `Edit`).

5. Faire plusieurs copies de la phrase, et essayer différentes déformations du signal, par exemple :
   – filtrer entre 0 et 1000 Hz, puis entre 1500 et 5000 Hz, en utilisant : `Filter/pass-band`
   – redresser le signal : `Modify/formula : 0.2*abs(self[col])`
   – le dichotomiser : `Modify/formula : if self[col]>0 then 0.2 else -0.2 endif`)
6. remplacer par un bruit blanc un autre segment de la phrase `intact.wav`, d'un des fichiers `ph*.wav` sur le site http ://www.pallier.org/misc/tp2/, ou d'une autre phrase que vous enregistrerez (Menu `New/Record mono sound`).

   Note : Utiliser les fonctions Cut/Copy/Paste de la fenêtre d'édition. Pour créer un bruit blanc : `New/Sound/Create Sound`, puis : `formula=randomGauss(0,0.1)`

3 Implications théoriques

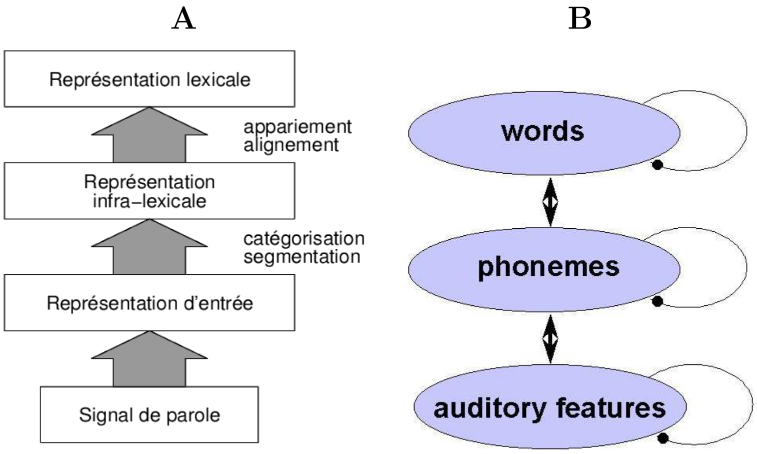
Dans les années 1940, avec les premières techniques d'enregistrement et de visualisation de la parole, il semblait qu'il allait devenir possible de définir des caractéristiques acoustiques invariantes associées à chaque son de parole (les phonèmes). Selon ce point de vue, la perception de la parole consisterait à identifier des patterns acoustiques correspondant, successivement, à chaque phonème. Une fois les phonèmes identifiés, les mots compatibles avec ceux-ci seraient recherché dans le « lexique mental ». Cela correspond à un modèle « montant » (*bottom-up*) de la perception de la parole (cf. figure 2A). Les grandes distortions qu'on peut imposer au signal de parole suggèrent que s'il existe des invariants, ceux-ci sont assez abstraits.

Des résultats comme ceux de Warren sont généralement interprétés en supposant que la perception est également influencée par les connaissances *a priori* du sujet. Dans les modèles dits « interactifs » McClelland and Elman [1986], McClelland and Rumelhart [1986], cette influence est due à un feed-back des niveaux supérieurs de traitements (lexique, sémantique,...) vers le niveau de décodage en phonèmes (figure 2B).

L'expérience originale de Warren (1970) montre-t-elle vraiment qu'il y a une influence des connaissances lexicales sur le décodage des phonèmes ?

Les connaissances des sujets influencent leurs décisions, mais il n'est pas évident qu'elles affectent l'étape précoce de décodage des phonèmes. En

Fig. 2 – Modèle montant (A) et interactifs (B) de la perception de la parole



d'autres termes, la réponse des sujets pourrait provenir d'un biais de réponse post-perceptuel.

Samuel [1981] a fait une remarque cruciale : les modèles interactifs prédisent une restauration phonémique plus importante dans des mots que dans des pseudomots, et il a utilisé l'approche de la *théorie de détection du signal* pour séparer les composantes perceptives et décisionnelles.

4 La théorie de détection du signal

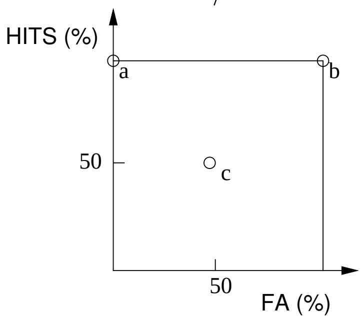
La théorie de détection du signal modélise la prise de décision.

Imaginez le scénario suivant : vous écoutez une série d'enregistrements bruités, et vous devez détecter, pour chacun d'entre eux, s'il contient un son cible fixé. Ce son est faible et la détection dans le bruit n'est pas évidente ; il y a donc quatre possibilités :

	Décision	
Cible présente	Non	Oui
Non	rejet correct	fausse alarme
Oui	manqué	détection correcte

À la fin de l'expérience, il y a donc 4 pourcentages, correspondant à chacune des cases du tableau. En fait, ces 4 pourcentages ne sont pas indépendants[2] : la performance d'une personne peut être caractérisée, par exemple, par le taux de détection correcte (Hits) et le taux de fausse-alarmes (FA). On peut la représenter par un point dans un repère FA/Hits. Par exemple, la figure 3 montre les performances de 3 sujets (points (a), (b) et (c)). (a) a une performance parfaite ; (c) répond au hasard et (b) répond systématiquement que le signal est présent (son taux de fausse alarme est de 100 %). Bien que les performances de (b) et de (c) soient différentes, on a envie de dire qu'ils ont la même *sensibilité* de détection (nulle), mais que leur *biais de réponse* sont différents. [3]

Fig. 3 – Diagramme fausses alarmes/détections correctes (FA/Hits)



Ces cas sont extrèmes. Grosso-modo, plus la performance d'un sujet est proche du point (a) (ou éloigné de la diagonale principale du carré, d'équation Hits=FA), plus sa sensibilité de détection est bonne. Un sujet non biaisé correspondra à un point situé sur la petite diagonale du carré (Hits=1-FA). La théorie de détection du signal fournit à partir des taux de FA et de Hits, des estimations de la sensibilité et du biais de réponse.

Suivant la théorie de la détection du signal Tanner and Swets [1954], le traitement perceptif d'un stimulus donne lieu à une réponse interne de l'organisme. La distribution de cette grandeur est différente selon que le stimulus contient la cible ou pas. Par exemple, la figure 4 décrit les distributions provoquées, chez un radiologue, par des clichés de patients ayant ou non des tumeurs. Sa prise de décision est modélisée par un seuil (figure 5)

Le calcul de la sensibilité (d') et du biais 'β (dans le modèle gaussien à variances égales), à partir des taux de fausse alarme et de hits, est assez simple :

$$d' = z(H) - z(F)$$

[2]Le bombre d'essais contenant la cible ou pas son fixés à l'avance. Par conséquent, la somme de spourcentage sur chaque ligne du tableau fait 100%.

[3]Remarquons que le biais de réponse peut varier au cours du temps chez le même sujet.

FIG. 4 – Distributions associées à un stimulus contenant ou non la cible (Modélisation d'une décision

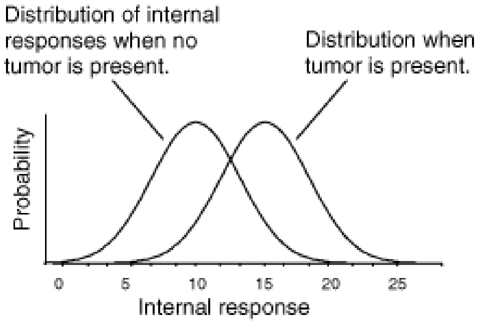
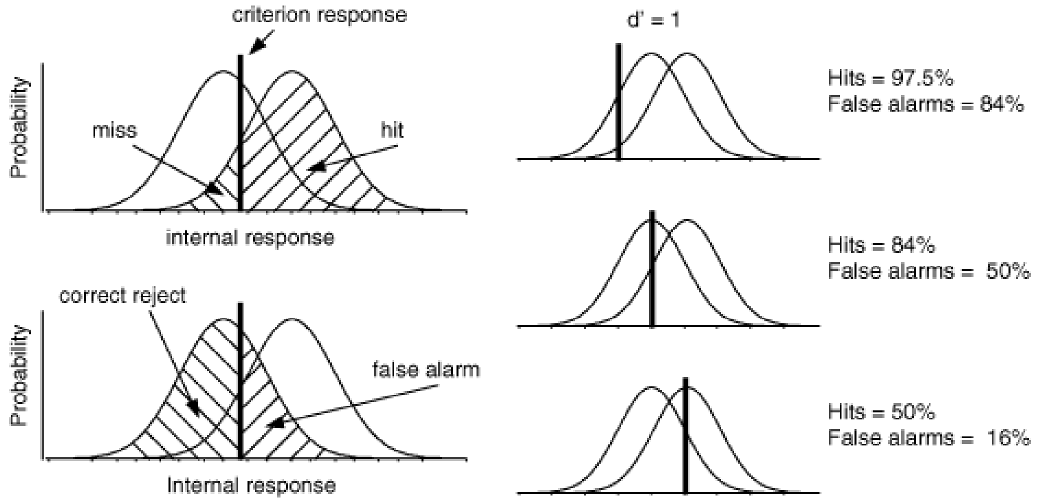


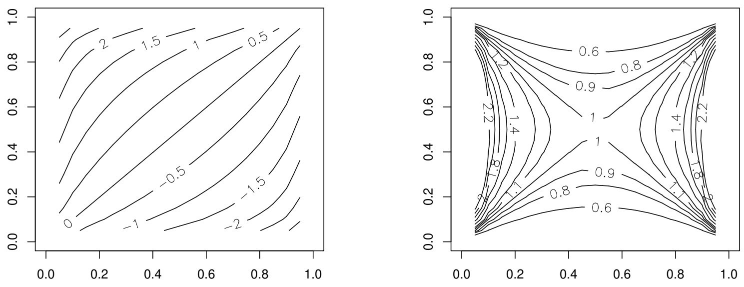
FIG. 5 – Effet de la position du critère de décision



$$\beta = e^{-\frac{z(H)^2 - z(F)^2}{2}}$$

Où z désigne la fonction inverse de la fonction de distibution de la loi normale. Autrement dit, z transforme un centile dans en “Z-score”.

FIG. 6 – Courbes d'iso-d' (à gauche) et d'iso-β (à droite



On peut calculer d' et β avec un tableur : il suffit de trouver la fonction inversant la distribution de la loi normale. Dans OpenOffice Calc, c'est la fonction LOI.NORMALE.STANDARD.INVERSE. Dans Excel, il semblerait que ce soit NORMSINV.

Certains logiciels de statistiques possèdent des fonctions permettant calculer d' et β. Dans le logiciel de statistique “R” (http ://www.r-project.org), on peut définir les fonctions :

```
dprime <- function(hit,fa) {
  qnorm(hit) - qnorm(fa) }

beta <- function(hit,fa) {
  zhr <- qnorm(hit)
  zfar <- qnorm(fa)
  exp(-zhr*zhr/2+zfar*zfar/2)
}
```

5 À vous de jouer : réalisation d'une expérience de détection de signal

Votre mission est de programmer une expérience très simple de détection d'un ton pur faiblement audible.

5.1 Construction des stimuli

Avec Praat, générer un bruit blanc de 1 seconde, puis 3 autres stimuli durant également une seconde, contenant un ton pur (c.-à.-d. une sinusoïde), superposé à un bruit blanc. Ajuster les amplitudes de la sinusoïde pour obtenir trois stimuli où le ton pur est difficilement audible (pour que la tâche de détection du ton pur ait un sens). Sauves ces sons comme des fichiers `bruit.wav, son1.wav, son2.wav, son3.wav.`

Solution : utiliser `New/Sound/Create Sound` et un formule du type :

```
0.1 * sin(2*pi*377*x) + randomGauss(0,0.1)
```

(Le facteur 0.1 n'est qu'un exemple.)

5.2 Programmation de l'expérience

Le sujet entend les stimuli, les uns après les autres, dans un ordre aléatoire. Il doit, après chaque item, indiquer s'il pense que le bruit a été ajouté ou s'il a remplacé le phonème critique.

Voici le code python minimal pour jouer un fichier son :

```
import pygame.mixer, pygame.time

pygame.mixer.init(22050)

sound=pygame.mixer.Sound('ph1.wav')
channel=sound.play()
while channel.get_busy():
     pygame.time.wait(10)
```

Pouvez-vous compléter ce code pour jouer une série de fichiers sons ?

Voici une possibilité :

```
import pygame.mixer, pygame.time

pygame.mixer.init(22050)

def play(file):
```

```
        sound=pygame.mixer.Sound(file)
        channel=sound.play()
        while channel.get_busy():
               pygame.time.wait(10)

liste=['ph1.wav','ph2.wav','ph3.wav','ph4.wav']

while (liste != []):
        stimulus = liste[0]
        liste = liste[1:]
        play(stimulus)
```

Notez que nous avons défini une fonction `play` qui joue un fichier son, et une boucle principale qui lit successivement les éléments de la variable `liste` et appelle la fonction play.

Il ne nous reste plus qu'à aléatoiriser l'ordre des stimuli et à enregistrer les réponses.

Pour aléatoiriser l'ordre des stimuli, il suffit d'ajouter, après la définition de la variable `liste` :

```
import random
random.shuffle(liste)
```

Pour détecter le bouton de souris qui est appuyé après chaque stimulus, on ajoute le code suivant dans la boucle principale :

```
        response = 0
        while not(response):
            for event in pygame.event.get():
                if event.type == MOUSEBUTTONDOWN:
                    response=event.button
```

Vous avez maintenant (presque) toutes les pièces du puzzle. Vous pouvez essayer de l'assembler vous-même. Quand ce sera fait, comparer votre résultat au regardez le code du script `signal.detection.py`. Il y a plusieurs façons de programmer la même chose...

6 Faire passer la manip et analyser les résultats

À partir du fichier de résultat, calculer les taux de hits et de fausse alarme (séparement pour son1, son2 et son3), puis calculer d' et β.

7 À vous de jouer : Réalisation d'une expérience semblable à celle de Samuel (1981)

Dans l'expérience 2 de Samuel [1981], les stimuli étaient des mots ou des pseudomots, dont un phonème avait été soit caché par du bruit, soit remplacé par du bruit. À chaque essai, le sujet entendait d'abord le stimulus original, intact, puis une des deux versions modifiées. Il devait dire si le phonème était juste caché ou remplacé par le bruit.

7.1 sélection du matériel

Sélectionner 20 mots français fréquents, bi ou trisyllabiques, contenant un son [f], [s] ou [ch]. (utiliser www.lexique.org)

Inventer 20 pseudomots similaires (mais pas trop ;-).

Enregistrer ces items avec Praat (`New/Record mono sound`).

7.2 Construction des stimuli

Pour chaque item, il faut construire une version où un bruit est *ajouté* sur le phonème critique, et une version ou le bruit *remplace* le phonème critique.

Pour additionner un bruit blanc sur un son dans un intervalle temporel, par exemple [0.693,0.776], dans Praat, utiliser `Modify/formula` :

```
if x>0.693 and x<0.776 then self[col]+randomGauss(0,0.1) else self[col] endif
```

Il se pose la question d'avoir un niveau de bruit adéquat (c.-à-d. tel que la tâche de détection ne soit ni trop facile, ni trop difficile). Voir l'article de Samuel (1981) : ce dernier ajuste le niveau de bruit à l'energie moyenne du signal remplacé (root mean square).

Il faudra peut-être piloter pour trouver les amplitudes adéquates.

7.3 Programmation de la manip

On pourrait utiliser le programme précédent `signal.detection.py`. Mais il faudrait construire à priori toutes les paires 'item_intact item_added' et 'item_intact item_replaced'. C'est fastidieux mais possible (et cela peut en fait être automatiser).

Une autre approche consiste à modifier la boucle principale du programme signal.detection.py pour qu'il joue le fichier intact avant le fichier modifié. Si le nom du fichier intact se deduit facilement du nom du fichier modifié, cette solution est rapide.

7.4 Faire passer l'expérience et regarder les résultats

8 Pour poursuivre

À propos des effets lexicaux sur la perception des phonèmes, le débat n'est pas clos : voir Norris et al. [2000], Samuel and Pitt [2003].

Ceux intéressés par la théorie de détection du signal pourront lire Macmillan and Creelman [2005].

Références

H. Fletcher. *Speech and Hearing.* Van Nostrand, New York, 1929.

J. C. R. Licklider. Effects of amplitude sitortion upon the intelligibility of speeh. *J. acoust. Soc. Amer.*, 18 :429–434, 1946.

J. C. R. Licklider. The intelligibility of amplitude dichotomized, time-quantized speech waves. *J. acoust. Soc. Amer.*, 22 :820–823, 1950.

Neil A. Macmillan and C. Douglas Creelman. *Detection theory : a user's guide.* LEA, 2nd edition, 2005.

J. L. McClelland and J. L. Elman. The trace model of speech perception. *Cognitive Psychology*, 18 :1–86, 1986.

J. L. McClelland and D. E. Rumelhart. *Parallel distributed processing : Explorations in the microstructures of cognition. Vol 2. Psychological and Biological Models.* MIT Press, Cambridge Mass., 1986.

G. A. Miller and J. C. R. Licklider. The intelligibility of interrupted speech. *Journal of the Acoustical Society of America*, 22 :167–173, 1950.

D. Norris, J. M. McQueen, and A. Cutler. Merging information in speech recognition : Feedback is never necessary. *Behavioral and Brain Sciences*, 23(3) : 299–370, 2000.

K. Saberi and D. R. Perrot. Cognitive restoration of reversed speech. *Nature*, 398 : 760, 1999.

A. G. Samuel. Phonemic restauration : Insights from a new methodology. *Journal of Experimental Psychology : General*, 110 :474–494, 1981.

A.G. Samuel and M.A. Pitt. Lexical activation (and other factors) can mediate compensation for coarticulation. *Journal of Memory and Language*, 48 :416–434, 2003.

W. P. Tanner and J. A. Swets. A decision-making theory of visual detection. *Psychological Review*, 61 :401–409, 1954.

R. M. Warren. Perceptual restauration of missing speech sounds. *Science*, 167 : 392–393, 1970.