Sommaire

1 Les modèles de langues neuronaux

1.1 Cas général

1.1.1 Construction de l'espace probabilisé

Notation : On note $A \mapsto |A|$ la fonction qui associe à un ensemble A son cardinal.

Définition 1.1 *On appelle vocabulaire un ensemble fini quelconque, noté $V = \{s_1, \dots, s_{|V|}\}$. Les s_i sont appelés symboles. On note ε le symbole vide qui n'appartient pas à V.*

Exemple de symboles :

— Un caractère

— Un mot

— Un bit

Exemple de vocabulaire :

— Ensemble des mots de la langue française

— Ensemble des caractères unicode

Définition 1.2 *Un texte T est un élément de V^L, où $L \in \mathbb{N}^*$.*

On cherche à définir une probabilité sur l'ensemble des textes. Définissons notre espace de probabilité.

Définition 1.3 *On appelle l'ensemble des textes $\Omega = \bigcup_{L=1}^{+\infty} V^L$. On note $\mathcal{A} = \sigma\left(\{\{T\} \| T \in \Omega\}\right)$, une tribu sur Ω.*

On note $L : T \in \Omega \mapsto |T|$ la variable aléatoire qui associe à un texte sa longueur. On définit les $(X_n)_{n\in\mathbb{N}^*}$ comme :

$$\forall i \in \mathbb{N}^*, X_i(T) = \begin{cases} i\text{-ème symbole de } T \text{ si } i \leqslant L(T) \\ \varepsilon \text{ si } i > L(T) \end{cases}$$

On suppose qu'il existe une probabilité $\mathbb{P}$ sur l'espace probabilisable $(\Omega, \mathcal{A})$. On dispose d'un échantillon de textes distribué selon la mesure $\mathbb{P}$ et on cherche à estimer $\mathbb{P}$ par une mesure de probabilité $\widehat{\mathbb{P}}$.

On appelle $\widehat{\mathbb{P}}$ un modèle de langue. En raison de la nature séquentielle du langage, on le construit en pratique en conditionnant sur les mots précédents du texte.

1.1.2 Construction d'un modèle de langue par probabilités conditionnelles

Soit un texte $T = s_1 \dots s_L \in V^L$, où $L \in \mathbb{N}^*$.
La probabilité d'observer T s'écrit :

$$\begin{aligned}
\mathbb{P}(T) &= \mathbb{P}\left(\bigcap_{i=1}^{L} X_i = s_i \cap \bigcap_{i=L+1}^{+\infty} X_i = \varepsilon\right) \\
&= \mathbb{P}\left(\bigcap_{i=1}^{L} X_i = s_i \cap X_{L+1} = \varepsilon\right) \text{ par construction des } X_i \\
&= \mathbb{P}(X_1 = s_1 \cap \ldots \cap X_L = s_L \cap X_{L+1} = \varepsilon) \\
&= \mathbb{P}(X_{L+1} = \varepsilon | X_1 = s_1, \ldots, X_L = s_L)\mathbb{P}(X_1 = s_1, \ldots, X_L = s_L) \\
&= \mathbb{P}(X_{L+1} = \varepsilon | X_1 = s_1, \ldots, X_L = s_L)\mathbb{P}(X_L = s_L | X_1 = s_1, \ldots, X_{L-1} = s_{L-1}) \times \\
&\qquad \mathbb{P}(X_1 = s_1, \ldots, X_{L-1} = s_{L-1}) \\
&= \prod_{i=1}^{L+1} \mathbb{P}(X_i = s_i | X_1 = s_1, \ldots, X_{i-1} = s_{i-1}) \text{ en posant } s_{L+1} = \varepsilon
\end{aligned}$$

Nous serons amenés à effectuer des approximations dans les calculs pour estimer ces probabilités. Ces différentes estimations conduisent à la définition de différents modèles de langues neuronaux.

Nous distinguons les modèles de langue suivants :

— les modèles n-grams

— les modèles Neural Network (NN)

1.2 Modèle n-gram

Dans un modèle n-gram, nous faisons l'hypothèse simplificatrice que la probabilité d'apparition du mot s_i ne dépend que de $n-1$ prédécesseurs. Ainsi,

$$\mathbb{P}(X_i = s_i | X_1 = s_1, \ldots, X_{i-1} = s_{i-1}) = \mathbb{P}(X_i = s_i | X_{i-(n-1)} = s_{i-(n-1)}, \ldots, X_{i-1} = s_{i-1})$$

— Cas $n = 1$: Modèle unigram : $\mathbb{P}(T) = \prod_{i=1}^{L+1} \mathbb{P}(X_i = s_i)$

— Cas $n = 2$: Modèle bigram : $\mathbb{P}(T) = \prod_{i=1}^{L+1} \mathbb{P}(X_i = s_i | X_{i-1} = s_{i-1})$

— Cas $n = 3$: Modèle trigram : $\mathbb{P}(T) = \prod_{i=1}^{L+1} \mathbb{P}(X_i = s_i | X_{i-2} = s_{i-2}, X_{i-1} = s_{i-1})$

— Cas $n > 3$: Modèle n-gram : $\mathbb{P}(T) = \prod_{i=1}^{L+1} \mathbb{P}(X_i = s_i | X_{i-(n-1)} = s_{i-(n-1)}, \ldots, X_{i-1} = s_{i-1})$

Nous estimons ces probabilités sur un corpus de textes et nous supposons que le corpus de textes reflète la langue dans l'absolu, ce qui sera le cas si nous disposons d'un très grand corpus de textes . Il s'agit là d'une approche statistique.

Etant donné que nous travaillons sur un corpus de textes fini, nous utilisons naturellement pour probabilité la mesure de comptage. Ainsi, le calcul de probabilité conditionnelle devient :

$$\mathbb{P}(X_i = s_i | X_{i-(n-1)} = s_{i-(n-1)}, \ldots, X_{i-1} = s_{i-1}) = \frac{|X_{i-(n-1)} = s_{i-(n-1)}, \ldots, X_{i-1} = s_{i-1}, X_i = s_i|}{|X_{i-(n-1)} = s_{i-(n-1)}, \ldots, X_{i-1} = s_{i-1}|}$$

Limitations : Etant donné que nous travaillons sur un corpus fini, nous avons une combinaison de mots finis. Il est possible qu'un mot qui n'apparaît pas dans le modèle. Sa probabilité d'apparition est donc nulle : $\mathbb{P}(X_k = s_k) = 0$. On parle de sparcité. Cette probabilité nulle pose problème : toute

séquence de mots qui n'apparaît pas dans le corpus a une probabilité égale à 0 d'apparaître. Notre modèle reconnaît donc uniquement des séquences connues.

Pour pallier ce problème et pouvoir généraliser à des séquences de mots non connues, nous pouvons effectuer un « lissage », qui consiste à attribuer une valeur de probabilité non nulle pour les mots n'apparaissant jamais dans le corpus.

1.3 Modèle Neural Network

Une approche permettant d'opérer un lissage des probabilités est l'utilisation de réseaux de neurones. Leur capacité à la généralisation leur permet de mieux estimer les probabilités de séquences rarement observées telles que les longues séquences où celles contenant des symboles rares. L'idée est de capturer les liens (ou caractéristiques) que les mots peuvent avoir entre eux. Ces liens sont représentés par les différentes connexions qui existent entre les neurones du réseau. On parle de « représentation distribuée ».

Un réseau de neurones, sous une forme simplifiée (modèle *feed-forward* basique), est une fonction formée de la composition de n fonctions de la forme :

$$f_i : X \mapsto \sigma_i(W_i \cdot X + b)$$

où $X \in R^{p_i}, p_i \in \mathbb{N}^*$; σ_i est une fonction non linéaire, appelée fonction d'activation (ReLU, tanh, sigmoid) ; W_i est une matrice de poids (apprise) et b un vecteur de biais (appris).

C'est donc une fonction continue de $\mathbb{R}^p$ dans $\mathbb{R}^q$, où $(p, q) \in \mathbb{N}^2$. Afin de l'utiliser comme estimateur de la probabilité conditionnelle d'un symbole sachant les précédents (i.e. pour en faire un modèle de langue), il faut représenter l'ensemble des symboles précédents comme un vecteur de R^p et les probabilités conditionnelles comme un vecteur de R^q. Les probabilités conditionnelles se représentent naturellement comme un vecteur de $R^{|V|}$ dont la somme des composantes fait 1.

La représentation de l'entrée est sujette à plusieurs méthodes :

- le *One-Hot Encoding* consiste à représenter chaque symbole précédent comme un vecteur de $R^{|V|}$ dont une composante vaut 1 et toutes les autres 0.
- les méthodes d'*embedding* consistent à associer à chaque mot un vecteur de R^p où $p \ll |V|$ de manière apprise.
- l'utilisation d'*embeddings* pré-appris (*fastText*, *GloVe*, *Word2Vec*) permet de créer une représentation statique (ne changeant pas pendant l'apprentissage).

1.4 Génération d'échantillons de texte

Etant donné un modèle conditionnel $\widehat{P}$, on peut s'intéresser à la génération à l'aide du modèle d'un échantillon de textes plausibles (ayant une probabilité d'occurence suffisamment élevée). La méthode de force brute, qui consiste à estimer une à une les probabilités de tous les textes d'une certaine longueur, est prohibitivement coûteuse en terme de calculs (coût exponentiel en la longueur).

Il existe diverses méthodes plus fines.

1.4.1 Méthode gloutonne

Une méthode naïve consiste, étant donné un échantillon initial $(s_1, \ldots, s_n)$, à procéder itérativement à la sélection du symbole ayant la probabilité d'occurence la plus élevée conditionnellement aux symboles précédents.

Formellement :

On se donne $L > n$ la longueur du texte à générer. A chaque étape i (i commençant à $n+1$) on sélectionne le symbole $s_i = \arg\max(\widehat{P}(s|s_1 \dots s_{i-1})|s \in V)$ jusqu'à ce que $i = L$, étape à laquelle l'algorithme termine.

En pseudo-code :

```
echantillon = [s1...sn]
for i in [n+1...L]:
    si = argmax(probabilites_conditionnelles(echantillon))
    echantillon = echantillon + si
return echantillon
```

1.4.2 Méthode Beam Search

Une méthode un peu plus évoluée que la méthode gloutonne consiste à garder en mémoire un ensemble de k échantillons pour finalement sélectionner le plus probable une fois arrivé à la longueur voulue.

Formellement :
On se donne $L > n$ la longueur du texte à générer. On se donne comme dans la méthode gloutonne un échantillon initial $(s_1, \dots, s_n)$. Le but est de constituer une famille de k échantillons de longueur L ainsi que leur probabilité conditionnelle à $(s_1, \dots, s_n)$: $[(T_1, P_1) \dots (T_k, P_k)]$. A la première étape, on prend la famille dégénérée $[(s_1 \dots s_n, 1) \dots (s_1 \dots s_n, 1)]$.

A chaque étape i (i commençant à $n+1$), on calcule pour chaque échantillon $T_j \in [T_1 \dots T_k]$ gardé en mémoire à l'étape précédente le vecteur de probabilités conditionnelles du symbole suivant. On multiplie ce vecteur par P_j pour obtenir la probabilité de l'échantillon complété par ce symbole. On dispose alors de $k|V|$ échantillons accompagnés de leur probabilité. On sélectionne les k plus probables pour obtenir le vecteur $[(T_1, P_1) \dots (T_k, P_k)]$.
on sélectionne le symbole $s_i = \arg\max(\widehat{P}(s|s_1 \dots s_{i-1})|s \in V)$ jusqu'à ce que $i = L$, étape à laquelle l'algorithme termine.

En pseudo-code :

```
echantillons = [[s1...sn],...,[s1...sn]]
probabilites = [1,...,1]
for i in [n+1...L]:
    for j in [1,...,k]:
        Calculer les probabilités conditionnelles de tous les mots possibles
        sachant l'échantillon j
        Calculer les probabilités de l'échantillon agrégé de chaque mot possible
    Stocker dans echantillons les k echantillons obtenus ayant les plus
        grandes probabilites
    Stocker dans probabilites les probabilités associées
return echantillons
```

1.4.3 Méthode de l'échantillonnage

Cette méthode consiste, étant donné un échantillon initial $(s_1 \dots s_n)$, à procéder itérativement à la sélection du symbole suivant en réalisant un tirage aléatoire selon les probabilités des symboles possibles conditionnellement aux symboles précédents.

Cette méthode est moins sensible à l'overfitting en évitant de générer systématiquement la même suite de symboles à partir d'un même contexte. Elle permet l'exploration en générant des séquences plus diverses que les méthodes précédentes, évitant notamment l'apparition de boucles infinies et de séquences apprises par coeur.

Formellement :

On se donne $L > n$ la longueur du texte à générer. A chaque étape i (i commençant à $n+1$) on sélectionne le symbole $s_i = \text{sample}(\widehat{P}(s|s_1 \dots s_{i-1})|s \in V)$ jusqu'à ce que $i = L$, étape à laquelle l'algorithme termine.

En pseudo-code :

```
echantillon = [s1...sn]
for i in [n+1...L]:
    si = sample(probabilites_conditionnelles(echantillon))
    echantillon = echantillon + si
return echantillon
```