TP 1 & 2 de Compression des Données

MULAPI TITA Ketsia

Comme à nos modestes habitudes, le présent travail se veut être une véritable reférence en terme de notions de compression, et ce, aussi bien du point de vu pratique que théorique, une amélioration de ce document est donc vivement encouragé.

A. INTRODUCTION

1. Les définitions des notions élémentaires

- Quid de la compression :

Compression, par définition il s'agit d'un processus qui permet de diminuer la taille des fichiers. ces fichiers peuvent être sous un format image, vidéo, musique, etc. (Wikipedia).

- Pourquoi faisons-nous de la compression ?

Pour résoudre le problème de stockage des données ainsi que leur traitement afin de faire circuler ces données avec une faible quantité de bits.

- Ce que nous allons faire :

Dans cette série de travaux pratique, nous allons à la découverte de quelques méthodes de compression (les plus populaires de l'histoire).

Ce travaille est principalement axé sur des méthodes de codages pour la compression à la source, qui servent à minimiser les messages transmis et stockés en tenant compte des propriétés de cette même source. Et donc, puisque compression = codage, notre objectif est de réaliser des codages sans perte, tout en optimisant le temps de calcul, dans le respect du principe de non création de l'information.

Néanmoins, nous notons que dans certains domaines le codage avec perte est acceptable, c'est le cas dans le domaine du signal et de l'image, tel que nous l'avons vu l'année dernière.

2. Illustration par un simple cas d'utilisation

Comme vu précédemment dans le cours, les statistiques textuelles sont une représentation en nombre, des propiétés d'un texte.

2.1. Le texte

Pour comprendre cela en plus simple, essayons d'illustrer ce que nous ferons plus loin, à l'aide d'un petit exemple : supposons que nous avons la phrase suivante *"**je travaille "*** on peut compter :

> 1 caractère "j"

> 2 caractères "e"

> 2 caractères " " (vide)

> 1 caractère "t"

> 1 caractère "r"

> 2 caractères "a"

> 1 caractère "v"

> 1 caractère "i"

> 2 caractères "l"

Et, c'est suite à cela que nous pourrions désigner la probabilité qui n'est rien d'autre que la fréquence relative à l'apparution de chaque caractère noté ici par petit << c >> dans la phrase selon que nous ferons pour chaque caractère c, le nombre de fois que ce caractère est présent dans le text, divisé par le nombre total des caractères dans le texte qui n'est rien d'autre que la taille du texte (Cfr. la loi des grand nombre) :

$$P(X = ci) = \frac{C}{n}$$

(exclusion mutuellement)

$$\forall\, i\ ,\ C = \ nombre\, de\, ci\, caractere\, et\, n = \ nombre\, total\, des\, caracteres\, (distincts)\, dans\, le\, texte$$

Aisni pour ***le caractère "j" la probabilité est de 1/13=0.07692*** alors que pour **"e" la probabilité est de 2/13=0.1538**, ce qui nous semble logique car il est plus probable de retrouve la lettre "e" dans notre texte, que la lettre "j". On dit que 13 est la taille de la phrase (message).

ainsi,

$$\sum_{i=1}^{n} P(X = ci) = \sum_{i=1}^{n} \frac{C}{n} = 1$$

qui n'est rien d'autre que la phrase (le texte, le message ou l'information).

Mais nous ne nous limiterons pas à ce stade, en effet, le but étant de compresser nous découvrirons un bon nombre de propriétés statistiques en plus de la distribution des probabilités, dans le point suivant.

2.2. Les propriété des informations à compressions

- 2.2.1) ***l'entropie de la source qui produit un message :*** qui n'est rien d'autre que l'information apportée en moyenne par un message. Dans un système où la distribution de cette entropie suit une loi gaussienne, on dit que mesurer la quantité d'information nécessaire pour décrire un message X est la borne inférieure du nombre de bits nécessaires en moyenne pour coder cette information, alors que la capacité du canal de transmission comme l'information mutuelle maximale (cas équiprobable) entre les messages, est la borne entropique supérieure des taux de transmission autorisés par ce canal.

elle peut s'écrire de la sorte :

$$H(T) = -\frac{1}{Longueur} \sum_{i=1}^{Longueur} log_2(p(ai))$$

ou

$$H(T) = -p_i \sum_{i=1}^{taille-alphabet} log_2(p(ai))$$

- 2.2.2) ***la perplexité :*** est une mesure très parlante de l'incertitude, elle nous permettra d'estimer le nombre de classes équiprobables qui produisent cette incertitude tout simplement parce que plus nous avons des données, plus il y a de forte chance (probable) d'avoir de la nuance. Et donc par exemple, avec la perplexité on pourra juger la qualité du modèle utilisé pour prédire de nouvelles données. On note :

$$PP(T) = 2^{H(T)}$$

- 2.2.3) ***la borne entropique :*** A permit à Shannone d'indiquer qu'il faut utiliser le minimum du minimum possible pour le codage de l'information, Par exemple si notre entropie est H=4.35bits, pourrions-nous arriver à un codage proche des 4.35 ? Non ! Ainsi la borne entropique ici n'est rien d'autre que le maximum, d'information que l'on peut atteindre. Elle s'exprime donc en fonction de la taille de l'alphabet :

$$BE = log_2(taille\, alphabet)$$

2. Conclusion

Et donc pour notre phrase *"**je travaille "***, nous avons les propiétés suivantes :

- alphabet = [' ' 'a' 'e' 'i' 'j' 'l' 'r' 't' 'v']
- Taill de l'alphabet = 9
- Distribution des caracteres = [1 2 2 1 1 2 1 1 1]
- Longueur du texte = 12
- Entropie (H) = 4.45 bits ou Shannon
- Perplexité = 21.9 classes
- Borne entropique = 3.16
- la première paire la plus fréquente (deux fois) est : "e "
- à la suite de cette mesure de relation entre symbole, la matrice des occurrences des bigrammes est :

```
[[0., 0., 0., 0., 0., 0., 0., 1., 0.],
 [0., 0., 0., 1., 0., 0., 0., 0., 1.],
 [1., 0., 0., 0., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 0., 1., 0., 0., 0.],
 [0., 0., 1., 0., 0., 0., 0., 0., 0.],
 [0., 0., 1., 0., 0., 1., 0., 0., 0.],
 [0., 1., 0., 0., 0., 0., 0., 0., 0.],
 [0., 0., 0., 0., 0., 0., 1., 0., 0.],
 [0., 1., 0., 0., 0., 0., 0., 0., 0.]]
```

N.B. : Ces mêmes traitements que nous ferons tout au long de ce document seront essentiellement effectué sur la célèbre histoire de Olivier Twist, il s'agit à la base d'un film américain réalisé par Frank Lloyd en 1922, adapté du roman de Charles Dickens Oliver Twist. Ici nous n'allons pas nous intérressé à l'histoire, nous nous contenterons juste de faire de la satistique textuelle📊. Plus loin, nous prendrons le soin de les interpréter avec des représentation graphique.

B. PREMIERE PARTIE - Calcul de statistiques textuelles

1. Choix du fichier "OliverTwist.txt" et analyse des propriétés textuelle de son contenu

Après chargement d'un texte en mémoire, on fait appel à la méthode unique( ) pour obtenir les occurrences des caractères, ainsi que l'alphabet de caractères utilisé dans le texte.

Nous commençons d'abord par par importer les librairies concernant cette première partie.

alphabet :
[' ' '!' '"' ',' '-' '.' '1' '2' '3' '4' '5' '6' ':' ';' '?' 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z' 'à' 'â' 'ç' 'è' 'é' 'ê' 'ë' 'î' 'ï' 'ô' 'ù' 'û']

Taill de l'alphabet : 79

Distribution des caracteres : [162517 1399 10379 13670 4190 7685 1 1 1 1 1 1 455 2532 1117 601 963 1086 351 636 307 310 194 775 472 14 876 1847 581 1984 375 300 306 754 418 107 350 9 77 4 1 55406 6625 18842 24740 103680 8286 6009 5717 54287 3839 629 39007 19656 50228 36638 18896 8441 46774 52156 52785 45833 13639 664 2686 2253 1685 4023 433 656 2325 11820 1643 1 365 12 475 378 407]

Longueur du texte : 919587

Dans cet exemple basé sur le code source fournit dans le document du TP, nous pouvons constater que notre premier fichier possède 80 symboles distincts, et a une taille de 942125 symboles.

Longueur du texte : 919587

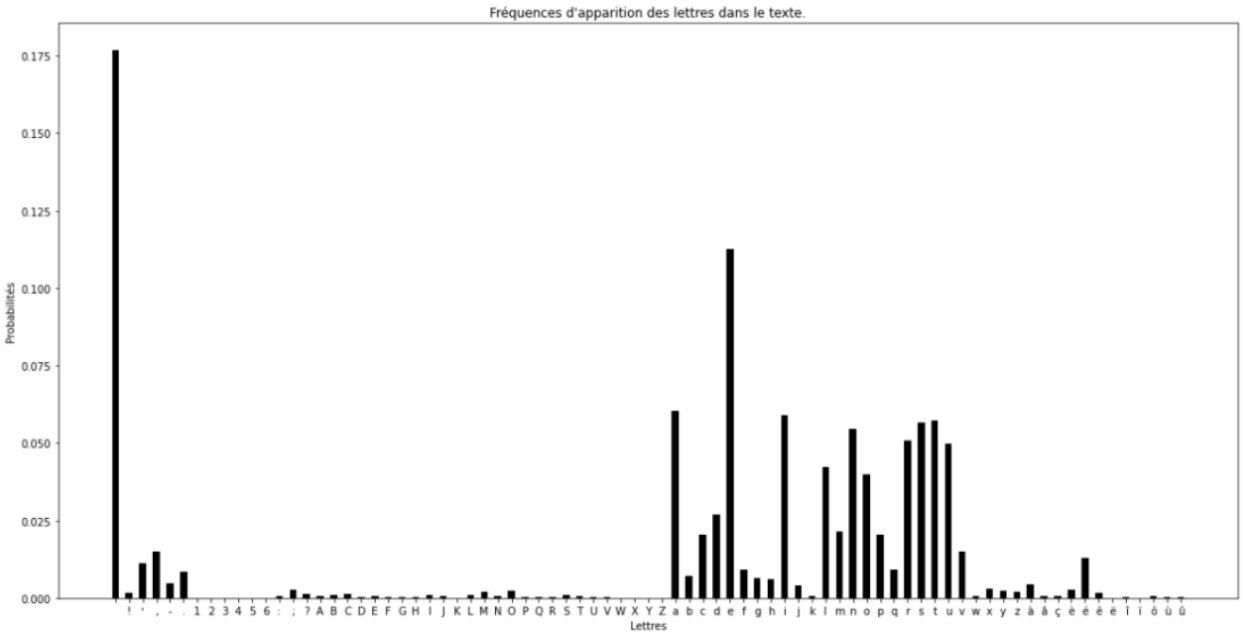
Dans cet exemple basé sur le code source fournit dans le document du TP, nous pouvons constater que notre premier fichier possède 80 symboles distincts, et a une taille de 942125 symboles.

Pour avoir une idée plus claire de ce que cela représente à l'aide de la probabilité, nous allons visuellement observer nos symboles dans leur entièreté :

- Probabilité (fréquence) :

la probabilité d'avoir petit << e >> dans notre texte est évaluée à : 0.1127462654430739

- Observons enfin l'histogramme de notre alphabet dans le texte, ainsi que les autres caractère par rapoort au symbole petit << e >> :



Nous pouvons constaté que le premier symbole qui est l'espace << >> , est plus présent que le symbole petit << e >>. Ce qui veut tout simplement dire qu'il est plus probable de retrouver, dans notre texte de l'espace que la lettre << e >>.

Nous pouvons constaté que le premier symbole qui est l'espace << >> , est plus présent que le symbole petit << e >>. Ce qui veut tout simplement dire qu'il est plus probable de retrouver, dans notre texte de l'espace que la lettre << e >>.

Calculer la distribution de probabilités des caractères (réalisé plus haut) et déduire l'entropie des textes, ainsi que leur perplexité, et la borne entropique, comparer ces valeurs à la taille de l'alphabet et au maximum d'entropie.

> ***L'entropie :***

Commentaire : Comme l'entropie représente à l'aide d'une fonction mathématique la quantité d'information transmise par la source, ici nous dirons que pour "OlivierTwist.txt", **4.4 bits ou Shannon** est la quantité moyenne d'information apportée à partir de la source.

Voici la représentation graphique de l'entropie de chaque symbole de notre texte

> ***La perplexité :***

Commentaire : Pour "OlivierTwist.txt" nous osons dire que par rapport à l'information que nous fournit l'entropie ci-haut, il existe **21.987907450322563** classes équipronbables qui produisent de l'incertitude

> La borne entropique :

Commenaire : Pour "OlivierTwist.txt", le maximum d'information que l'on peut atteindre est de **6.321928094887363**

2. Réalisons à présent ce même procédé sur les fichiers suivant, afin d'examiner ensemble les statistiques textuelles de ces fichiers :

1 - 'OliverTwistChap1.txt'

2 - 'OliverTwistChap2.txt'

3 - 'OliverTwistChap3.txt'

Chapitre	Entropie H	Perplexité	Longueur Texte	Taille alphabet	Borne entropique
1	4.32	20.06	2830	59	5.88
2	4.36	20.51	23503	68	6.08
3	4.40	21.20	6194	61	5.93

3. CONCLUSION DE CETTE PREMIERE PARTIE :

Dnas ce tableau récapitulatif, on peut constater que l'entropie des chapitres 3 (4.40) et 2 (4.36) sont les plus proche, cela revient à dire que leur contenu sont presque semblable. Et c'est aussi le cas avec la taille de leur alphabet respectif.

Le chapitre 3 est le plus perplexe et donc présente beaucoup plus de classes avec une incertitude bien précise.

Enfin, alors que dans le cahpitre 1 on peut atteindre au maximum 5.88 qté d'inforamtion, au cahpitre 2 et 3 c'ets plutot 6.08 et 5.93, c'est la Borne entropique.

C. DEUXIEME PARTIE - La méthode Byte Paire Encoding (PBE)

1. Présentation de la méthode BPE

Nous avons vu dans le cours comment cette première méthode de Shanone qui concenre le codage de chaque symbole a(i) de notre message, pris (symbole) individuellement a démontré ses limites car, une question capitale se pose en ce qui est du Codage de Shannone Fano, qu'en est-il de la mémoire ? doit-on toujours, coder un seul caractère (symbole) avec un bit où n'est-il pas préférable de faire des traitements sur des couples de symboles (Bigramme, Trigrammes, etc.) pou un même bit par exemple ? Ceci constitue l'object de la deuxième partie. Nous parlons alors des méthodes statistiques bigrammes.

Il s'agit d'une méthode très simple de codage de l'information qui consiste par itérations successives à coder par un seul symbole la paire de symboles la plus fréquente dans la source qu'on souhaite compresser. Cette méthode a été proposée pour la première fois dans la référence suivante : Philip Gage, "A New Algorithm for Data Compression", in the C Users Journal, 1994. en 1994

2. Calcul des statistiques textuelles bi-grammes

Toujours avec le même fichier, "OliverTwist.txt" trouvons la paire de symbole pour laquelle le nombre d'occurence est maximal. Nous avons les propiétés suivantes :

alphabet :
[' ' '!' '"' ',' '-' '.' '1' '2' '3' '4' '5' '6' ':' ';' '?' 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z' 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z' 'à' 'â' 'ç' 'è' 'é' 'ê' 'ë' 'î' 'ï' 'ô' 'ù' 'û']

Taill de l'alphabet : 79

Distribution des caracteres : [162517 1399 10379 13670 4190 7685 1 1 1 1 1 1 455 2532 1117 601 963 1086 351 636 307 310 194 775 472 14 876 1847 581 1984 375 300 306 754 418 107 350 9 77 4 1 55406 6625 18842 24740 103680 8286 6009 5717 54287 3839 629 39007 19656 50228 36638 18896 8441 46774 52156 52785 45833 13639 664 2686 2253 1685 4023 433 656 2325 11820 1643 1 365 12 475 378 407]

Longueur du texte : 919587

La paire, bigramme dans ce fichier, est **"e "** qui revient à **35148 fois** dans le fichier OlivierTwist.txt

Conclusion, la paire la plus fréqeunte dans ce fichier est **"e "** qui revient 35148 fois.

A est appellée la amtrice des occurences des bigrammes.

3. Enfin dans cette deuxième partie nous avons décidé de travailler avec le fichier "OliverTwist2" (ce choix est du au fait qu'il présente la plus petite quantité de texte, ce qui nous permettra d'effectuer nos traiement rapidement), ce fichier présente les caractéristiques suivantes :

alphabet :
[' ' '!' '"' ',' '-' '.' '/' ':' ';' 'A' 'C' 'D' 'E' 'H' 'I' 'L' 'M' 'O' 'P' 'Q' 'R' 'S' 'T' 'U' 'a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'l' 'm' 'n' 'o' 'p' 'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z' 'à' 'â' 'ç' 'è' 'é' 'ê' 'ï' 'î' 'ô' 'ù' 'û']

Taill de l'alphabet : 59

Distribution des caracteres : [480 1 32 36 1 9 4 2 2 1 1 3 1 2 3 1 8 3 2 3 1 3 1 157 19 70 90 348 26 20 10 172 10 114 59 166 107 83 35 141 143 170 159 37 2 10 5 1 9 2 2 11 38 3 1 1 4 2 3]

Longueur du texte : 2830

4. Inteprétation des résultats de compression obtenus à l'aide de la méthode BPE

4.1. Visualisation individuelle des propriétés

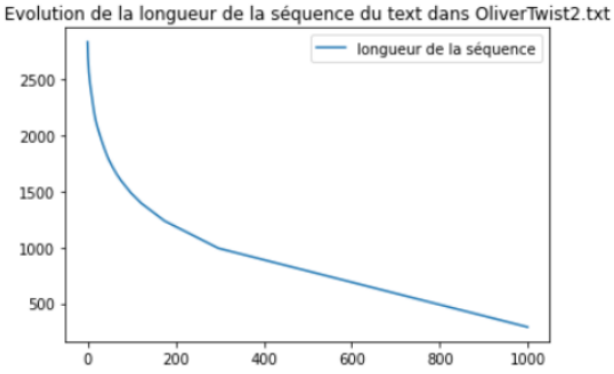


Figure 1 : Nous avons ici une idée de ce qui se passe pendant la compression, Nomre d'itération sur les abscisses, de caractères sur les ordonnées, il s'agit compression ou diminition progressive de la taille de notre fichier jusqu'à l'obtention d'un caractère représentatif.

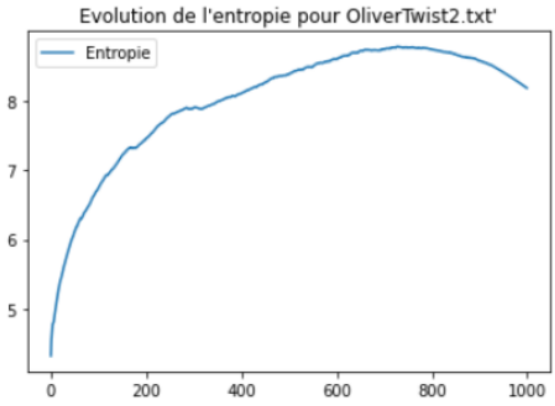


Figure 2 : Comme vous pouvez le constater, autant nous compressons, autant nous avons de l'incertitude dans l'information compressé.

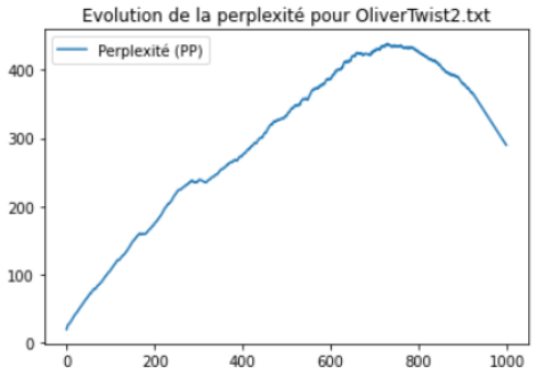


Figure 3 : Ici contrairement à l'entropie, il s'agit des moments pendants les quels on a pu observer une incertitude très parlante (exacte) au moment de la compression. On évalue à quel les stades les plus perplexe au moment de la compression.

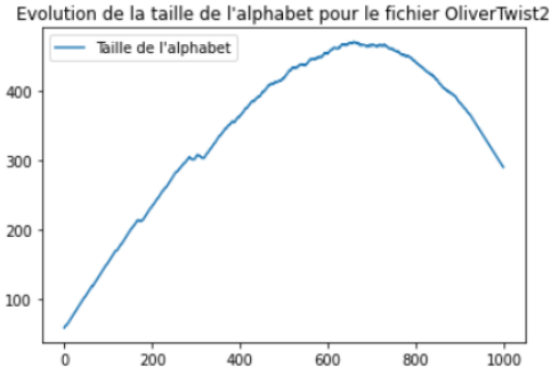


Figure 4 : Plus on compresse, et plus la taille de l'alphabet compressé augmente.

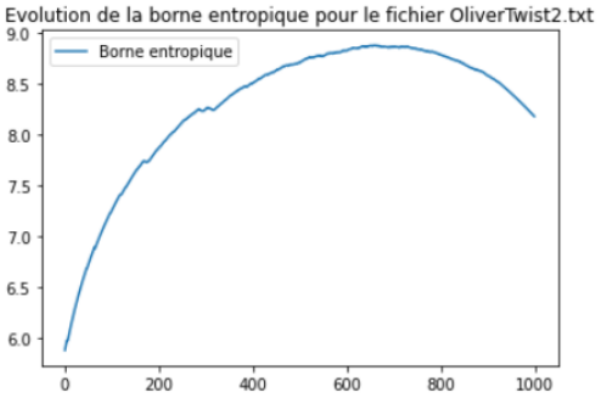


Figure 5 : La quantité d'information maximale qui pouvait être atteint au momeent de la compression.

4.2. Visualisation parrallèle des propriétés

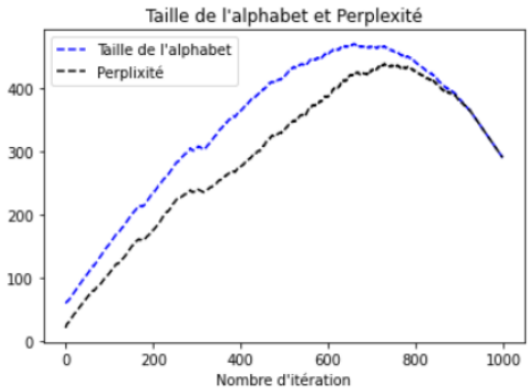


Figure 6 : Au moment de la compression, lorsque nous rencontrons d'avantage de contenu alphabétique, cela a une incidence considérable sur la perplexité l'information compréssée.

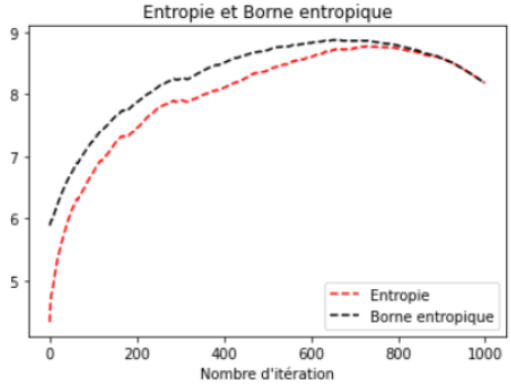


Figure 7 : Enfin ce qui nous intérresse tant, c'est de dire qu'ici nous avons une représentation gaussienne des valeurs entropiques et, que l'entropie au fur et àmesure de la compression, atteint la borne maximale (cas équiprobable) de la Borne entropique, pour ensuite se confondre et décoître ensemble au fur et à mesure que nous compressons les données.

Une autre méthode similaire à BPE est << WordPiece >> dont l'algorithme se présente de la sorte :

Algorithm

1. Prepare a large enough training data (i.e. corpus)
2. Define a desired subword vocabulary size
3. Split word to sequence of characters
4. Build a languages model based on step 3 data
5. Choose the new word unit out of all the possible ones that increases the likelihood on the training data the most when added to the model.
6. Repeating step 5until reaching subword vocabulary size which is defined in step 2 or the likelihood increase falls below a certain threshold.

source : https://medium.com/@makcedward/how-subword-helps-on-your-nlp-model-83dd1b836f46

6. CONCLUSION DE CETTE DEUXIEME PARTIE :

Le but de ce TP était de faire de la stastistiques textuelle et, de la compression unigramme et bigramme. Alors que pour le premier nous nous sommes basées uniquement sur des approches statistique, nous avons pour le second utilisé la méthode de BPE.

Les deux méthodes étant jugés moins optimal au regard de le performance en temps de calcul sur des petit quantité de données, laisse à croire qu'elle seront plus lourde à prendre en main sur des plus grandes quantités de données. Autre constat, c'est le non retour, comment faire pour décompressé les données ? Selon le repsonsable du Cours, cela constituera l'objet du prochain TP qui.