Projet IAS 2020-2021

RENCIOT Antoine
ALLOUANE Yohann
BENBOUZIANE Adame
BERRAMDANE Salah-Eddine
ICIL Bilal
POISSON Guillaume

Contents

1 Objectif

L'objectif de notre projet est de pouvoir reconnaître une langue selon un texte. Comme il existe plusieurs alphabets selon les langues, nous avons décidé de nous concentrer sur les langues utilisant l'alphabet latin. Nous allons comparer deux méthodes, la première est la méthode des N-grammes et la seconde sera un SVM multiclasse.
Dataset: `https://www.kaggle.com/zarajamshaid/language-identification-datasst`

2 Tâches

2.1 Exploration du dataset

Le dataset est assez simple, un texte avec sa langue. Il y a 22 langues dans le dataset, mais comme nous nous concentrons sur l'alphabet latin nous allons en garder 11. Les textes font en moyenne 150 caractères et chacune des langues ont 1000 textes.

2.2 Prétraitement des textes

Le nettoyage des textes est assez simple nous allons convertir toutes les lettres vers leur équivalent latin, puis nous allons supprimer tous les caractères spéciaux.

2.3 N-Gramme

Prétraitement:

Pour les features, nous allons utiliser la méthode des n-grammes et plus particulièrement la méthode des n-grammes lettres. Nous allons choisir la méthode 3-grammes. Il faut donc que chacun de nos textes ait les occurrences de tous les 3-grammes, 2-grammes et 1-grammes du texte.

Méthodes:

Pour la classification des textes, nous allons utiliser la méthode n-gramme qui est une méthode de classification généralisée. Nous allons prendre le cas des 3-grammes. L'hypothèse est que dans une séquence c de k caractères, la probabilité de l'apparition du caractère à la position i dépend des deux caractères à la position $i-1$ et $i-2$.

$$P(c_i|c_1, ..., c_{i-1}) = P(c_i|c_{i-2}, c_{i-1})$$

On a pour la probabilité d'un texte de k caractères sachant que le texte est de langue l :

$$P(c_1, ..., c_k|l) = P(c_1|l) \times P(c_2|c_1, l) \times P(c_3|c_1, c_2, l) \times ... \times P(c_k|c_{k-2}, c_{k-1}, l)$$

Simplifié en :

$$P(c_1, ..., c_k|l) = P(c_1|l) \times P(c_2|c_1, l) \prod_{i=3}^{k} P(c_i|c_{i-2}, c_{i-1}, l)$$

Pour calculer la probabilité d'un 3-gramme nous avons :

$$P(c_3|c_1, c_2, l) = \frac{P(c_1c_2c_3|l)}{P(c_1c_2|l)} = \frac{P(c_1c_2c_3|l)}{\sum_{x \in A} P(c_1c_2x|l)}$$

Où A est l'alphabet. Un problème se pose lorsqu'un 3-gramme n'apparaît pas dans le corpus d'entraînement, la probabilité de la séquence est alors 0. Pour pallier ce problème, nous allons appliquer un lissage additif ou lissage de Laplace [1]. On va prétendre que chaque 3-grammes a été vu une fois et on va ajuster les comptes en conséquence.

Pour prédire la langue d'un texte, nous n'avons plus qu'à choisir la langue dont la probabilité est la plus haute.

$$l^* = argmax(P(texte|l) \times P(l))$$

$$P(l) = \frac{|C_l|}{|C|}$$

Où $P(l)$ est le nombre de textes dans le corpus de langue l sur le nombre de textes de tous les corpus.

Un autre problème se pose, les probabilités tendent à être très petite de l'ordre de 10^{-355}, il faut rendre la formule plus stable numériquement. Nous allons passer sous forme log, l'argmax ne sera pas changé.

$$l^* = \arg\max(log(P(texte|l) \times P(l)))$$
$$l^* = \arg\max(log(P(texte|l)) + log(P(l)))$$

$$log(P(c_1, ..., c_k|l)) = log(P(c_1|l) \times P(c_2|c_1, l) \prod_{i=3}^{k} P(c_i|c_{i-2}, c_{i-1}, l))$$

$$log(P(c_1, ..., c_k|l)) = log(P(c_1|l)) + log(P(c_2|c_1, l)) \sum_{i=3}^{k} log(P(c_i|c_{i-2}, c_{i-1}, l))$$

Pipeline:

Pour commencer, nous allons préparer les données, pour un nombre de données n on prend 70% de données d'entraînement et 30% de données de test. Pour les données de test, chaque donnée aurait sa langue et son texte. Pour les données d'entraînement on compte les 3-grammes, 2-grammes et 1-grammes sur chaque corpus de langue.

Pour chaque langue, on aura donc un modèle qui sera les occurrences des

3-grammes, 2-grammes, 1-grammes qui vont nous permettre de calculer les probabilités des textes. On a plus qu'à appliquer chacun des modèles sur nos données d'entraînement et de test, le modèle ayant la plus haute probabilité sera notre prédiction. On fait cela sur plusieurs itérations où la taille du dataset varie.

2.4 SVM multi-classe

Prétraitement:

Pour cette méthode les features seront les fréquences des 3-grammes dans le texte. On va prendre les 100 3-grammes les plus fréquents.

Méthodes :

Un modèle SVM est un classifieur discriminatif et il peut être multi-classe [2], on va chercher une matrice $\boldsymbol{W} \in \mathbb{R}^{\mathcal{L} \times d}$ (où $\mathcal{L}$ sont les langues et d est le nombre de features) tel que le score de la langue du texte soit plus élevé que le score des autres langues. Il faut que l'écart entre le score de la vraie langue du texte et le deuxième meilleur score soit au moins d'une marge m . Le corpus d'entraînement est de taille n avec les entrées notées $\boldsymbol{x} \in \mathbb{R}^d$ les sorties notées $\boldsymbol{y} \in \mathbb{R}^{\mathcal{L}}$ sont représentées par des vecteurs hot-one.

$$\forall 1 \leq i \leq n, \boldsymbol{y} \neq \boldsymbol{y}^{(i)} \quad \boldsymbol{y}^\top \boldsymbol{W} \boldsymbol{x}^{(i)} + m \leq \boldsymbol{y}^{(i)\top} \boldsymbol{W} \boldsymbol{x}^{(i)}$$

La fonction à minimiser par rapport au paramètre $\boldsymbol{W}$ est la somme des mauvaises classifications par la marge :

$$J(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{W}) = \sum_{i=1}^{n} \max(0, \max_{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}} (y^\top \boldsymbol{W} \boldsymbol{x}^{(i)}) + m - \boldsymbol{y}^{(i)\top} \boldsymbol{W} \boldsymbol{x}^{(i)})$$

$$\nabla_{\boldsymbol{W}} J = \begin{bmatrix} \frac{dJ}{d\boldsymbol{W}_1} \\ \dots \\ \frac{dJ}{d\boldsymbol{W}_{|\mathcal{L}|}} \end{bmatrix}$$

$$\begin{aligned} \frac{d}{d\boldsymbol{W}_l} J &= \frac{d}{d\boldsymbol{W}_l} \sum_{i=1}^{n} \max(0, \max_{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}} (\boldsymbol{y}^\top \boldsymbol{W} \boldsymbol{x}^{(i)}) + m - \boldsymbol{y}^{(i)\top} \boldsymbol{W} \boldsymbol{x}^{(i)}) \\ &= \sum_{i=1}^{n} \frac{d}{d\boldsymbol{W}_l} \max(0, \max_{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}} (y^{\top (i)}) + m - \boldsymbol{y}^{(i)\top} \boldsymbol{W} \boldsymbol{x}^{(i)}) \end{aligned}$$

$$a = \max_{\boldsymbol{y} \neq \boldsymbol{y}^{(i)}} (\boldsymbol{y}^\top \boldsymbol{W} \boldsymbol{x}^{(i)}) + m - \boldsymbol{y}^{(i)\top} \boldsymbol{W} \boldsymbol{x}^{(i)}$$

$$\frac{d}{d\boldsymbol{W}_l} \max(0, a) = \begin{cases} \frac{d}{d\boldsymbol{W}_l} a & \text{si} \quad a > 0 \\ 0 \quad \text{sinon.} \end{cases}$$

$$
\begin{aligned}
\frac{d}{d\boldsymbol{W}_l}a &= \frac{d}{d\boldsymbol{W}_l}(\max_{\boldsymbol{y}\neq\boldsymbol{y}^{(i)}}(\boldsymbol{y}^{\top(i)}) + m - \boldsymbol{y}^{(i)\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) \\
&= \frac{d}{d\boldsymbol{W}_l}(\max_{\boldsymbol{y}\neq\boldsymbol{y}^{(i)}}(\boldsymbol{y}^{\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) + m - \boldsymbol{y}^{(i)\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) \\
&= \frac{d}{d\boldsymbol{W}_l}\max_{\boldsymbol{y}\neq\boldsymbol{y}^{(i)}}(\boldsymbol{y}^{\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) - \frac{d}{d\boldsymbol{W}_l}\boldsymbol{y}^{(i)\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) \\
&= \frac{d}{d\boldsymbol{W}_l}\boldsymbol{z}^{\top}\boldsymbol{W}\boldsymbol{x}^{(i)} - \frac{d}{d\boldsymbol{W}_l}y^{(i)\top}\boldsymbol{W}x^{(i)}) \quad \text{où} \quad \boldsymbol{z} = \arg\max_{\boldsymbol{y}\neq\boldsymbol{y}^{(i)}}(\boldsymbol{y}^{\top}\boldsymbol{W}\boldsymbol{x}^{(i)}) \\
&= \boldsymbol{z}^{\top}\boldsymbol{x}^{(i)} - \boldsymbol{y}^{(i)\top}\boldsymbol{x}^{(i)}
\end{aligned}
$$

$$
\nabla_{\boldsymbol{W}}J = \sum_{i=1}^{n}\begin{cases} \boldsymbol{z}^{\top}x^{(i)} - \boldsymbol{y}^{(i)\top}\boldsymbol{x}^{(i)} & \text{si} \quad a > 0 \\ 0 \quad \text{sinon.} \end{cases}
$$

Pour trouver notre paramètre $\boldsymbol{W}$ tel que

$$
\arg\min\sum_{i=1}^{n}\max(0, \max_{\boldsymbol{y}\neq\boldsymbol{y}^{(i)}}(\boldsymbol{y}^{\top}\boldsymbol{W}x^{(i)}) + m - \boldsymbol{y}^{(i)\top}\boldsymbol{W}x^{(i)})
$$

On réalise une descente de gradient :

$$
\boldsymbol{W} \mapsto \boldsymbol{W} - \eta\nabla_{\boldsymbol{W}}J
$$

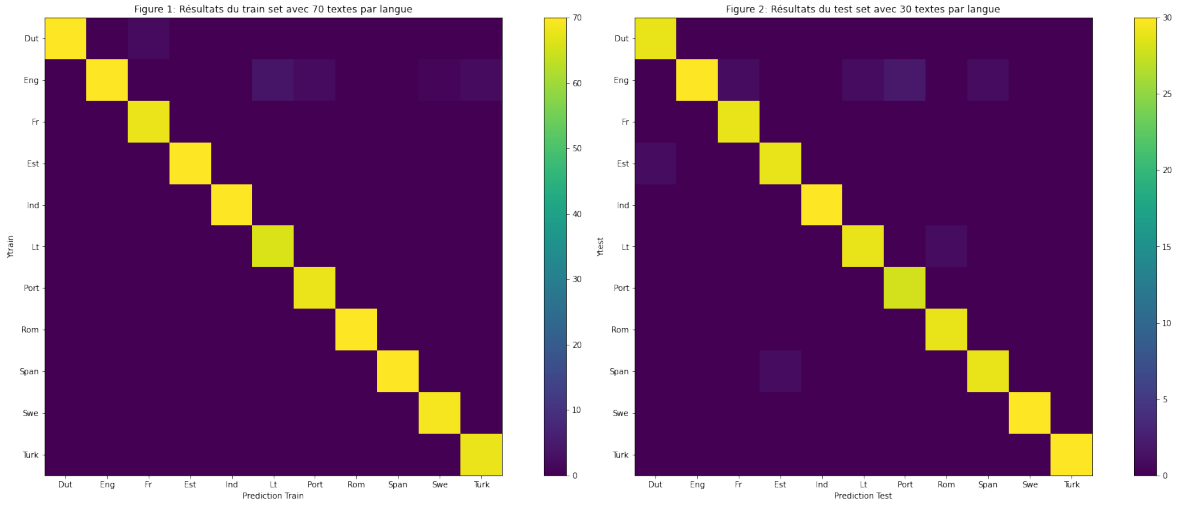
Pipeline

Même chose que pour les N-grammes nous allons préparer les données. Pour les données d'entraînement et de test, on prend 70% et 30%. Pour chaque texte, on calcule la fréquence des 3-grammes puis on prend le top 100 des 3-grammes les plus fréquents. Ce top 100 sera nos features.

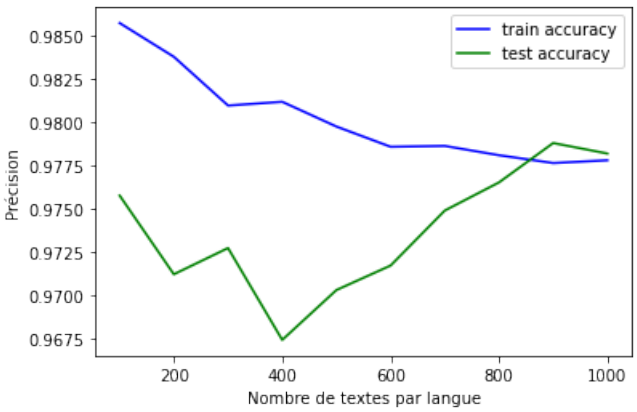
Nous allons ensuite, avec une matrice $\boldsymbol{W}$ initialisée aléatoirement et nos données d'entraînement, réaliser une descente de gradient pour trouver la matrice où notre taux d'erreurs est le plus bas. On applique cette matrice sur notre de données de test. On fait cela sur plusieurs itérations où la taille du dataset varie.

3 Résultats

3.1 N-grammes

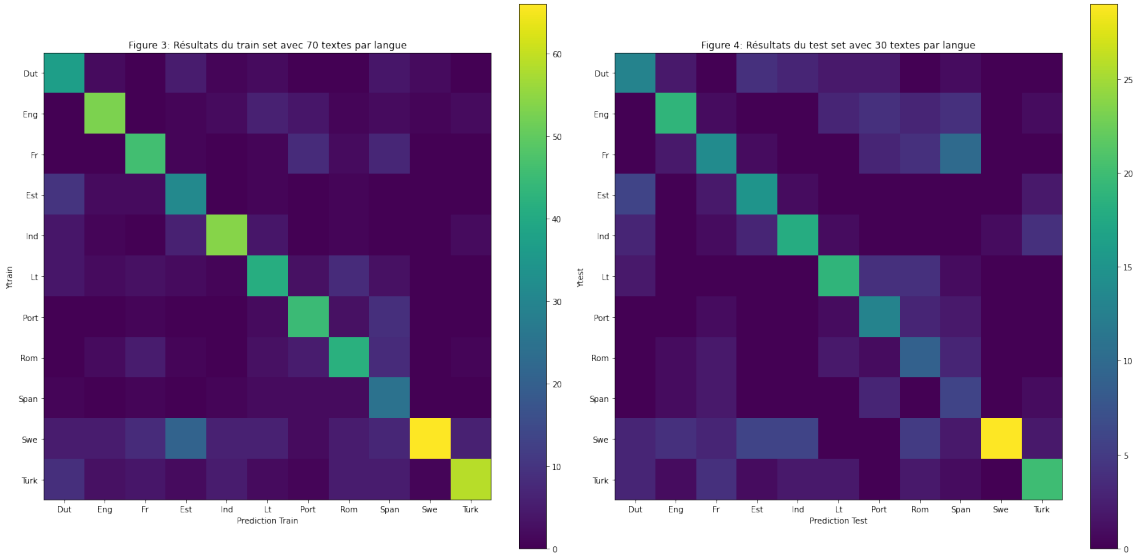


Pour cette méthode, les résultats sont assez bons avec un petit dataset. On a une précision d'environ 97% pour le test set avec 30 textes par langue.

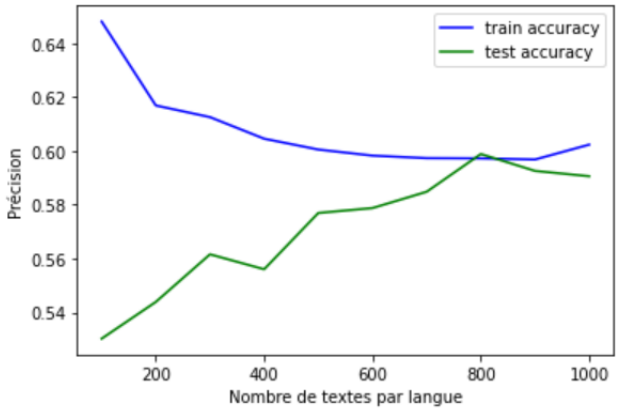


Les résultats ne varient pas énormément en fonction de la taille du data set. La méthode N-gramme semble une méthode très efficace pour ce qui est de la reconnaissance des langues. De plus l'écart entre le train set et le test set est assez minime environ 1%, ce qui nous laisse à penser qu'il n'y a pas d'overfitting.

3.2 SVM multi-classe



Pour le SVM multi-classe, les résultats sont moins bons que la méthode N-gramme. Une précision d'environ 50% avec un test set de 30 textes par langue. On remarque que les textes suédois ont des bons résultats.



Lorsque la taille du dataset augmente, la précision de la prédiction augmente de quelques pourcents. Vers 800 textes, la courbe semble se stabiliser, il aurait peut-être fallu plus de texte pour en être sûr. Le choix des 100 3-grammes les plus fréquents est aussi peut-être un choix peu judicieux. En effet, même si nous avons choisi les 100 3-grammes les plus fréquents, beaucoup de textes avaient des fréquences de 3-grammes fixés à 0 ce qui rend la classification moins efficace.

On avait remarqué avant que les textes suédois ont plutôt de bonnes prédictions. Le choix de features à peut-être grandement avantagé la classification des textes suédois.

Pour cette méthode, l'écart entre le train set et le test set tend à se réduire. L'overfitting n'a pas l'air d'être présent.

4 Conclusion

Pour classifier des textes, on aura tendance à choisir la méthode des N-grammes plutôt que le SVM. Avec la méthode N-gramme, nous n'avons pas besoin d'optimiser un paramètre et en plus de cela, elle donne des résultats assez satisfaisant sur un petit dataset.

5 Références

References

[1] Lissage additif - Additive smoothing `https://fr.qaz.wiki/wiki/Additive_smoothing`

[2] Caio Corro - [TC1] Notes on Machine Learning `http://teaching.caio-corro.fr/2020-2021/TC1//tc1.pdf`