Université de Paris

université PARIS DIDEROT PARIS 7

DÉTECTION DE LA LANGUE D'UN TEXTE

Projet de TAL L3 LI – 2020-2021

Sous la direction
De Madame Bingzhi li

Réalisé par
Théo Falgayrettes
Loubna Khennou

Table des matières

Liste des tableaux

Liste des figures

Introduction

La fréquence d'apparition d'un caractère est un indice statistique qui diffère d'une langue à une autre. Le diagramme ci-dessous nous montre qu'effectivement, il existe une dissimilitude de la fréquence des lettres dans 11 langues, ceci sans parler de l'existence des caractères non alphabétiques tels que les symboles de ponctuation, les parenthèses, les accolades ou bien les symboles mathématiques qui ne se présentent pas de la même manière dans toutes les langues. De ce fait, pour pouvoir détecter la langue d'un texte, il sera primordial d'analyser les segments du texte et prendre en compte le maximum de cas possible.

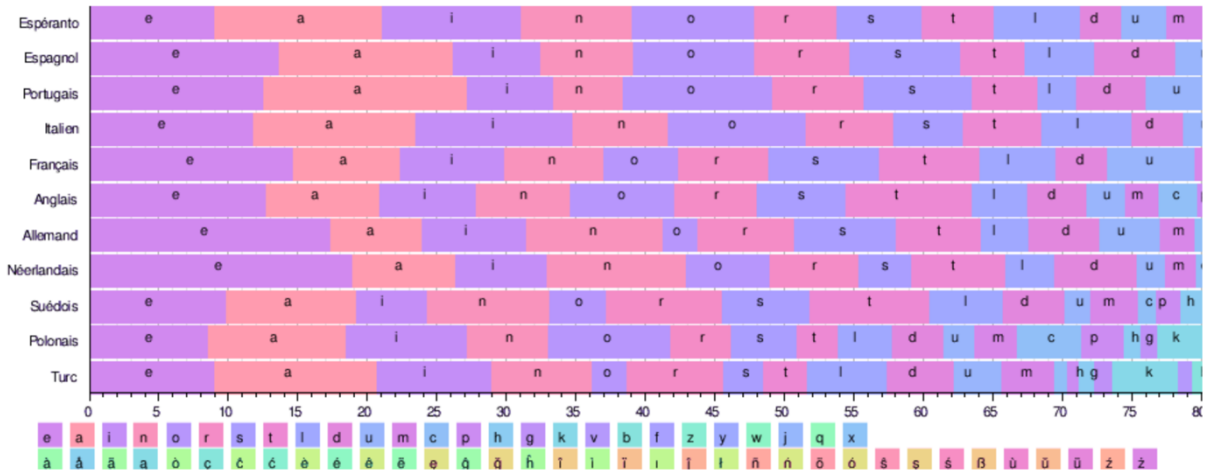

Figure 1: Diagramme comparatif de la fréquence des lettres dans 11 langues[1]

Dans le cadre du projet du fin du L3, nous avons choisi de travailler sur le thème de la détection automatique de la langue d'un texte. Ce travail s'inscrit dans le cadre de la linguistique-informatique : deux domaines qui possèdent de grandes masses de textes, ce qui entraine l'usage de différents modèles de langues probabilistes.

Durant ce projet, nous allons avoir recours au modèle n-grammes, en particulier les modèles unigramme, bigramme et trigramme. L'objectif est d'estimer les probabilités de ces n-grammes pour une langue dans un corpus de textes. A partir de ces estimations, nous visons à évaluer la performance de chaque modèle et sa capacité à détecter la langue de chaque texte faisant partie de notre corpus. C'est ainsi pour ces raisons que nous avons choisi de travailler sur ce projet car il va nous permettre de mettre en pratique nos connaissances fondées en salle de cours tout au long de cette année.

[1] https://fr.wikipedia.org/wiki/Fréquence_d%27apparition_des_lettres_en_français

Pour atteindre nos objectifs, nous commencerons en premier lieu par une description générale puis spécifique du sujet tout en exposant la méthode suivie pour réaliser les taches à automatiser. En deuxième lieu, nous explorons les données utilisées, les différentes expériences réalisées sans oublier la méthode d'évaluation du système en présentant les résultats de cette évaluation. En troisième lieu, nous présenterons d'une part la structure de notre programme sous forme d'une brève description simple et précise et d'autres part les choix d'implémentation accompagné des problèmes que nous avons rencontrés et les solutions que nous avons proposées. Et en dernier lieu, nous clôturons ce travail un manuel utilisateur qui porte sur les indications pratiques pour exécuter le programme.

1. Partie théorique

La langue représente un système de caractères qui permet la communication entre les personnes. La reconnaissance de la langue se base principalement sur la détection de caractères dans les phrases du texte et l'analyse de leurs fréquences nous permettant ainsi de se rapprocher d'une langue.

Durant ce projet, nous souhaitons identifier la langue d'un texte donné. Pour cela, nous utilisons une base de données de textes dont la langue est préalablement déterminée. Dans un premier lieu, l'objectif est de construire des modèles caractérisant les différentes langues, basés sur la fréquence d'apparition de caractères dans le texte de chaque langue. Dans un deuxième lieu, le but est de comparer et analyser la performance de chaque modèle par rapport à la longueur du texte, afin de déterminer le modèle adéquat qui permet la détection de la langue d'un texte cible.

Avant d'entamer le sujet, nous avons commencé tout d'abord par choisir les langues sur lesquelles nous voulons travailler. Nous sommes donc partis sur : le français, l'anglais, l'espagnol et l'italien. En effet, nous nous sommes focalisés sur les langues qui ont presque les mêmes alphabets pour faciliter à la fois la création des dictionnaires et le traitement des caractères (n-grammes) de chaque texte par la suite.

La réalisation de ce projet a nécessité le suivi de trois étapes qui consistent en l'apprentissage, la prédiction et l'évaluation qui viendra dans le chapitre suivant.

1.1. Apprentissage

Cette étape consiste à constituer un corpus d'apprentissage à partir des langues choisies dans le but d'estimer une base de données afin de traiter le texte cible et détecter sa langue. Pour chaque texte, nous avons suivi trois modèles :

- Unigramme consiste en l'occurrences d'un seul caractère dans le texte.
- Bigramme repose sur l'occurrences de deux caractères dans le texte.
- Trigramme comprend l'occurrences de trois caractères dans le texte.

La construction de chaque modèle de langue est basée sur la méthode d'identification qui est fondée sur l'observation de la fréquence d'apparition du 1-gramme, bi-gramme et trigramme dans le texte. Par exemple, un texte français comportera beaucoup plus de [e] qu'un texte rédigé en anglais ou en italien.

En effet, cette modélisation correspond au modèle de Markov d'ordre n où uniquement les n dernières observations sont utilisées pour la prédiction de la lettre suivante. Ainsi le modèle bigramme est un modèle d'ordre 2.

Nous notons que lors de la construction de ces modèles nous avons pris en considération tous les caractères du texte y compris la ponctuation et les espaces blanc car à la différence du français par exemple, l'anglais n'a pas d'espace avant les deux points, les points d'exclamation ou d'interrogation.

1.2. Prédiction

Pour constituer notre base de données et prédire la langue du texte cible, nous avons créé des dictionnaires pour chaque corpus de langue afin de stocker les différentes occurrences de caractères des textes. Autrement dit, pour chaque corpus nous parcourons les phrases, nous les décomposons en séquences de n caractère selon le modèle choisi et nous stockons leurs occurrences dans un dictionnaire. Ces dictionnaires constitueront par la suite nos dictionnaires de référence que nous allons incrémenter au fur et à mesure des tests pour rendre plus riche notre base de données sur laquelle nous allons s'appuyer pour reconnaitre la langue d'un texte.

Dans un autre sens, pour pouvoir mesurer la similarité entre le texte cible et nos dictionnaires de référence, nous avons séparé le texte en phrases. Ensuite, nous avons créé un dictionnaire pour chaque phrase à partir des modèles choisis. Ceci nous a permis d'avoir une liste de dictionnaires où chaque dictionnaire est réalisé à partir de la fréquence d'apparition des unigrammes, bigrammes ou bien des trigrammes pour toutes les phrases du texte cible.

Ayant créé une liste de dictionnaires pour chaque phrase en se basant sur les modèles donnés, nous avons mesuré la similarité de chaque dictionnaire de la liste réalisée avec les dictionnaires de référence préalablement créé des quatre langues choisies. C'est ainsi la raison pour laquelle nous avons eu recours au calcul de la formule du cosinus θ qui se présente comme suivant :

$$\cos\theta = \frac{x^T y}{\|x\|\|y\|} \quad (1)$$

Conséquemment, plus $\cos\theta$ est proche du 1, plus les n-grammes sont proches les uns des autres et plus $\cos\theta$ se rapproche du 0, plus les n-grammes du texte cible sont loin de ceux du dictionnaire de référence d'une langue donnée.

Grace à cette formule, nous avons renvoyé une liste de cosinus par langue que nous avons comparé les unes par autres pour voir, pour chaque phrase, vers quelle langue elle se rapproche le plus.

A travers ces résultats, nous avons pu détecter la langue du texte, en revanche il existe une question qui demeure primordiale à poser. Elle consiste en l'évaluation de la performance des modèles. Autrement dit, bien que nous ayons réussi à se rapprocher de la langue du texte, il nous reste à mesurer la performance des modèles l'un par rapport aux autres pour pouvoir déceler le modèle le plus efficace lors de la détection de la langue.

2. Partie expériences et résultats

2.1. Description des données utilisés

Les données textuelles dont il est question dans ce projet sont multiples en termes de quantités. Un premier regroupement de données concerne celles présentées sur le site du Universal Dependencies[2] qui proposent 104 langues. Parmi ces données, on a choisi quatre corpus de référence d'un même texte de langues différentes :

- Es_pud-ud-test.txt pour l'espagnol.
- It_ pud-ud-test.txt pour l'italien.
- En_pud-ud-test.txt pour l'anglais.
- Fr_pud-ud-test.txt pour le français.

[2] https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3687

Contrairement au premier regroupement, le deuxième concerne les données à analyser, décelées à partir du même site pour pouvoir déterminer la langue. Pour cela, nous avons sélectionné quatre corpus qui désignent un texte distinct pour chaque langue donnée :

- Es_gsd-ud-test.txt pour l'espagnol.
- It_ partut-ud-test.txt pour l'italien.
- En_partut-ud-test.txt pour l'anglais.
- Fr_ partut-ud-test.txt pour le français.

Pour rendre plus clair notre démarche de regroupement de données, nous avons créé un tableau qui rassemble nos corpus et énumère la taille de chacun d'eux en termes du nombre de symboles/caractères existant

Langues	**ITA**	**ESP**	**ENG**	**FR**
Taille du corpus de référence	125592	124436	111914	128525
Taille du nouveau texte	19940	20004	18504	13873

Tableau 1: Information sur les corpus

2.2. Description des expériences réalisés et la méthode d'évaluation du système

Dans le cadre de la troisième étape de ce projet, et pour évaluer les trois modèles, nous avons effectué des tests en mettant en avant la longueur du texte cible. Ceci en tronçonnant automatiquement le corpus de test en n caractères. Plus précisément, nous avons eu recours à plusieurs test sets de taille 20 jusqu'à 6668 (phrases) et de longueur 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 50, 75, 100, 200, 500, 1000, 2000 ou 3000 caractères par langue.
Conséquemment, suivant tous les modèles sur lesquels nous avons travaillé, nous avons mesuré la similarité de chacune des phrases de n n-grammes avec le corpus de référence.
L'intérêt est de désigner combien de phrases sont détectées ayant n caractères sur l'ensemble des phrases dans tout le corpus à analyser.
Par ailleurs, l'élaboration de ces tests nous a servi également à déterminer le score d'accuracy qui indique le pourcentage de bonnes prédictions de la langue d'un texte, permettant par la suite de mesurer la performance globale des modèles.
En effet, pour trouver l'accuracy, nous avons énuméré les phrases détectées dans le corpus par les modèles et nous avons appliqué la formule :

$$Le\ score\ d'accuracy = \frac{le\ nombre\ de\ phrases\ détéctées\ dans\ le\ corpus\ par\ un\ modèle}{le\ nombre\ de\ phrase\ total\ du\ corpus} * 100 \quad (2)$$

C'est en se basant sur cet accuracy que nous allons choisir le modèle le plus performant par la suite. En d'autres termes, plus le score s'approche du 100%, plus le modèle est performant et inversement, plus il s'éloigne du100%, moins c'est la performance du modèle.

2.3. Traitement des résultats de l'évaluation des modèles

Pour rapporter plus concrètement les résultats de ce projet, nous avons converti nos aboutissements en tableaux et graphes de façon à ce que les différentes sorties du système soient plus perceptible.

Taille de la phrase (en caractères)	3	4	5	6	7	8	9	10	11	12
Nb de phrases	6646,7	4985	3988	3323,3	2848,6	2492,5	2215,6	1994	1812,7	1661,7
Unigrammes (%)	99,969 91124	99,959 87964	99,974 92477	99,969 91576	99,9648 9996	99,9598 8769	100	100	100	100
Bigrammes (%)	100	100	100	100	100	100	100	100	100	100
Trigrammes (%)	99,969 91124	100	100	100	100	100	100	100	100	100

Taille de la phrase (en caractères)	13	14	15	20	25	50	75	100	200	500	1000	2000	3000
Nb de phrases	1533,8	1424,3	1329,3	997	797,6	398,8	265,9	199,4	99,7	39,9	19,9	10,0	6,6
Unigrammes (%)	100	100	100	100	100	100	100	100	100	100	100	100	100
Bigrammes (%)	100	100	100	100	100	100	100	100	100	100	100	100	100
Trigrammes (%)	100	100	100	100	100	100	100	100	100	100	100	100	100

Tableau 2 : Résultats pour la langue italien

Conformément à ce qui ressort du tableau, les tests réalisés sur le corpus italien nous permettent d'avoir une quantité différente de phrases tout dépend du nombre de caractères définissant leurs tailles. En effet, pour 3 caractères, le nombre de phrase dans le texte est de 6646. De ce fait le modèle le modèle bigramme est à 100%, alors que les autres modèles unigramme et trigramme reçoivent un score d'accuracy qui va jusqu'au 99,96991124%. En sus, pour 4, 5, 6, 7, et 8

caractères, le nombre total de phrase varie de 4985 à 2492,5. Dans ce cas les modèles bigramme et trigramme arrive à détecter la langue de toutes les phrases du corpus italien ce qui fait un score d'accuracy de 100%. En revanche, l'unigramme reste moins performant par rapport aux autres modèles. Son accuracy varie de 99,95987964% jusqu'à 99,97492477%. Et pour 9, 10, 11, 13, 14, 15, 20, 25, 50, 75, 100, 200, 1000, 2000 et 3000 caractères, le nombre de phrases ayant ces tailles et entre 2215,6 et 6,6. Cette division de caractère permet de détecter toutes les phrases du corpus, chose qui aboutit à un score d'accuracy de 100% pour les trois modèles : unigramme, bigramme et trigramme.

Taille de la phrase (en caractères)	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Nb de phrases	6668	5001	4000,8	3334	2857,7	2500,5	2222,7	2000,4	1818,5	1667	1538,8
Unigrammes (%)	99,82003599	99,84003199	99,85003749	99,880024	99,86004199	99,84006397	99,82006298	99,85007496	99,89004948	99,880024	99,74009097
Bigrammes (%)	99,77504499	99,84003199	99,85003749	99,85002999	99,79006298	99,64014394	99,82006298	99,85007496	99,78009896	99,880024	99,80506823
Trigrammes (%)	99,86502699	99,82003599	99,92501875	99,880024	99,79006298	99,84006397	99,86504723	99,95002499	99,89004948	99,880024	99,87004548

Taille de la phrase	**15**	**20**	**25**	**50**	**75**	**100**	**200**	**500**	**1000**	**2000**	**3000**
Nb de phrases	1333,6	1000,2	800,2	400,1	266,7	200,0	100,0	40,0	20,0	10,0	6,7
Unigrammes (%)	99,85007496	99,8001998	99,87515605	99,75062344	100	100	100	100	100	100	100
Bigrammes (%)	99,77511244	99,9000999	99,75031211	99,75062344	100	100	100	100	100	100	100
Trigrammes (%)	99,92503748	100	99,87515605	100	100	100	100	100	100	100	100

Tableau 3 : Résultats pour la langue espagnol

Le tableau ci-dessus révèle les résultats des tests effectués sur le corpus espagnol. Nous notons que pour les phrases constituées de 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25 ou 50 caractères, leur nombre total passe de 6668 à 400,1 phrases. A cet effet, les trois modèles n'aboutissent pas à un score d'accuracy de 100% mais plutôt à un pourcentage d'environ 99% sauf pour le modèle trigramme qui arrive à détecter toutes les phrases ayant une taille de 20 et

50 caractères. Il s'agit donc du modèle le plus performant parmi les 13 divisions de caractères qu'on a réalisées.

Cependant, au-delà de 75 jusqu'à 3000 caractères, toutes les phrases du corpus ont été reconnu par les trois modèles. Ceci dit que l'accuracy atteint un pourcentage de 100% pour l'unigramme, le bigramme et le trigramme.

Taille de la phrase (en caractères)	3	4	5	6	7	8	9	10	11	12	13
Nb de phrases	6168	4626	3700,8	3084	2643,4	2313	2056	1850,4	1682,2	1542	1423,4
Unigrammes (%)	99,90272374	99,91353221	99,8919211	99,93514916	99,96217852	99,91353221	99,90272374	99,8919503	99,88116459	99,87029831	99,85955056
Bigrammes (%)	99,80544747	99,82706442	99,8388165	99,80544747	99,81089259	99,82706442	99,80544747	99,8392545	99,8174688	99,80544747	99,85955056
Trigrammes (%)	99,88651102	99,91353221	99,8919211	99,93514916	99,92435703	99,9567661	99,90272374	99,9497515	99,9058229	99,93514916	99,92977528

Taille de la phrase	15	20	25	50	75	100	200	500	1000	2000	3000
Nb de phrases	1233,6	925,2	740,2	370,1	246,7	185,0	92,5	37,0	18,5	9,3	6,2
Unigrammes (%)	99,91896272	99,89200864	99,8650472 3	100	100	100	100	100	100	100	100
Bigrammes (%)	99,83792545	99,89200864	99,8650472 3	100	100	100	100	100	100	100	100
Trigrammes (%)	99,91896272	100	100	100	100	100	100	100	100	100	100

Tableau 4 : Résultats pour la langue anglaise

Le tableau montre la sortie du système pour le corpus de la langue anglaise. Effectivement, pour les phrases ayant 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20 ou 25 caractères, on obtient un nombre de phrases qui varie entre 6168 et 740,2. En plus, comme pour les résultats du texte espagnol, hormis le modèle unigramme et trigramme qui enregistre un score de 100% respectivement pour les phrases constituées de 14 caractères (unigramme), 20 et 25 caractères (trigramme), les phrases ayant 50, 75, 100, 200, 500, 1000, 2000 et 3000 ont été toutes détectées sur l'ensemble du corpus anglais, ce qui explique le pourcentage du score d'accuracy qui ne baisse pas en delà de 100% .

Taille de la phrase (en caractères)	3	4	5	6	7	8	9	10	11	12	13
Nb de phrases	4624,3	3468,3	2774,6	2312,2	1981,9	1734,1	1541,4	1387,3	1261,2	1156,1	1067,2
Unigrammes (%)	99,28648649	99,22167772	99,31531532	99,26502378	99,2431887	99,19308357	99,35149157	99,42363112	99,44532488	99,22212619	99,25093633
Bigrammes (%)	99,84864865	99,85586624	99,89189189	99,87029831	99,89909183	99,82708934	99,87029831	99,92795389	99,9207607	99,91356958	99,90636704
Trigrammes (%)	99,95675676	99,884693	99,89189189	99,87029831	99,89909183	99,88472622	99,87029831	99,92795389	99,9207607	99,91356958	99,90636704

Taille de la phrase	15	20	25	50	75	100	200	500	1000	2000	3000
Nb de phrases	924,9	693,7	554,9	277,5	185,0	138,7	69,4	27,7	13,9	6,9	4,6
Unigrammes (%)	99,24324324	99,42363112	99,45945946	99,64028777	99,45945946	100	100	100	100	100	100
Bigrammes (%)	99,89189189	100	100	100	100	100	100	100	100	100	100
Trigrammes (%)	99,89189189	100	100	100	100	100	100	100	100	100	100

Tableau 5 : Résultats pour la langue française

Le tableau ci-dessous nous illustre les résultats que nous avons trouvé à l'issue des tests réalisés sur le corpus français. Nous remarquons que les phrases ayant 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 50 ou 75 caractères ne sont pas toutes détectées par le modèle unigramme, contrairement aux deux autres modèles bigramme et trigramme qui, quant à eux, ont une performance meilleure surtout pour les phrases composées de 15, 15, 20, 25, 50 et 75 caractères ou symboles (100%). En ce qui concerne la suite des tests à savoir les phrases de 100, 200, 500, 1000, 2000 et 3000 caractères, les trois modèles marquent une performance d'accuracy avantageuse de 100%.

Par ailleurs, nous avons essayé de rassembler tous ces résultats en un seul tableau pour pouvoir par la suite comparer aisément les trois modèles utilisés, tout en partant du principe qu'on ne connait pas la langue du texte à analyser au préalable.

Taille de la phrase (en caractères)	3	4	5	6	7	8	9	10	11	12	13
Unigrammes (%)	99,74478936	99,73378039	99,75804967	99,76252817	99,75757729	99,72664186	99,76856957	99,7914141	99,8041347 4	99,7431121 2	99,71264446
Bigrammes (%)	99,85728528	99,88074066	99,89495276	99,88144394	99,87501185	99,82357442	99,87395219	99,9039885 7	99,8806516 3	99,8997602 6	99,89274646
Trigrammes (%)	99,9195515	99,9045653	99,9272079 3	99,9213678 7	99,9033779 6	99,9203890 8	99,9095173 2	99,9559885 1	99,9378481 2	99,9321856 8	99,92654695

Taille de la phrase (en caractères)	15	20	25	50	75	100	200	500	1000	2000	3000
Unigrammes (%)	99,75307023	99,77895989	99,79991569	99,8477278	99,8648 6486	100	100	100	100	100	100
Bigrammes (%)	99,87623245	99,94802713	99,90383984	99,9376 5586	100	100	100	100	100	100	100
Trigrammes (%)	99,9339 7302	100	99,9687 8901	100	100	100	100	100	100	100	100

Tableau 6 : La moyenne des résultats pour tous les corpus

L'illustration de nos tests sous forme de moyenne nous a permis d'apercevoir les scores moyens d'accuracy pour les trois modèles. Comme le montre le tableau, pour le modèle unigramme, les phrases de taille 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 50 ou 75 caractères ne sont pas complètement détecté en moyenne, à l'opposé de celles constituées de 100, 200, 500, 1000, 2000 et 3000 caractères. Pareillement pour le modèle bigramme qui rajoute aux phrases détectées celles ayant 75 caractères. Quant au modèle trigramme, l'intervalle des phrases reconnues s'élargie davantage. Celui-ci arrive à détecter en moyenne aussi bien toutes phrases de 20 caractères que celles de 50, 75, 100, 200, 500, 1000, 2000 et 3000 caractères.

2.4. Analyse des résultats obtenus

Les résultats ci-dessus témoignent l'évaluation des modèles utilisés lors de ce projet pour la détection de la langue d'un texte cible. Nous notons en premier lieu que le modèle trigramme et bigramme demeurent les plus performants parmi les trois modèles. En effet, au fur et à mesure que nous accroissons le nombre de caractères de la phrase, leur score d'accuracy ne cesse d'augmenter. En revanche, bien que le modèle unigramme nous permet de détecter toutes les phrases d'un corpus quand celles-ci contiennent plus de 100 caractères, il reste moins performant quand la longueur de la phrase est moindre. Par conséquent, la détection de la langue d'un texte dépend essentiellement de l'efficacité du modèle dont nous faisons appel et de la longueur du texte à analyser. En d'autres termes, plus le modèle n-grammes a une échelle élevée des grammes, plus l'analyse sera concise et la langue reconnue sera appropriée au texte cible.
Le graphique ci-dessous illustre scrupuleusement cette relation entre la longueur du texte et la performance des modèles.

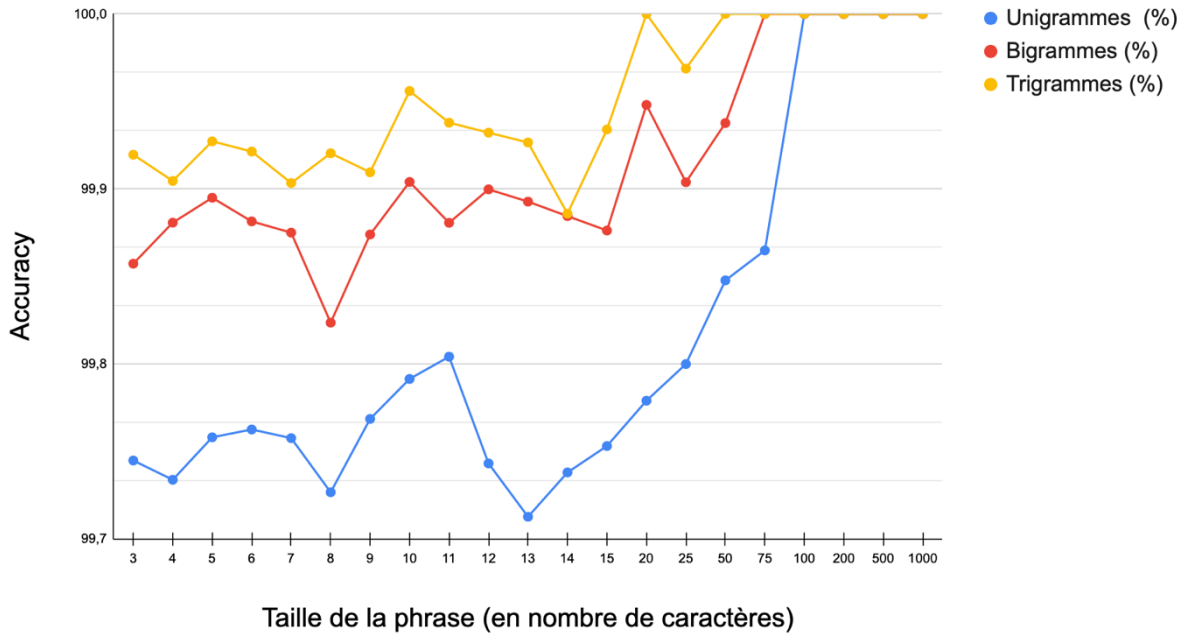

Figure 2 : Accuracy moyenne par taille de corpus

Dans un second lieu, nous avons décelé à travers les résultats que le taux d'identification global de la langue est moyennement faible voire n'aboutit pas à 100% jusqu'au 75 caractères par phrase. Ceci peut être dû, ou bien du fait que les langues déterminées sont principalement confondues, ou bien du fait que les données ne sont pas forcément représentatives car les tests réalisés n'ont pas été fait sur un corpus énorme. De ce fait, nous jugeons que le regroupement

de données pourrait être plus optimal avec un choix large de langues et un corpus de texte considérable.

3. Partie informatique

3.1. Description de la structure du Code

GdocData.py : prend en paramètre une liste de listes : 2 fonctions, une pour récupérer des données de notre Google sheet3 et la deuxième pour mettre à jour le Google sheet.

CodeProjet.py : programme principal

Processus d'analyse d'un modèle *:*

a. Chargement des dictionnaires de référence
b. Nouveau text (texte à analyser) = file_open() : se référer au fichier Readme pour connaître les modalités d'utilisation
c. Trois fonctions comptant les n-grammes : 1 pour la création d'un dictionnaire d'unigrammes, une pour les bigrammes et une pour les trigrammes
d. Calcul_cos() = calcul du cos entre le dictionnaire d'un texte et un dictionnaire de référence
e. text_tokenized(new_text) = séparation du corpus en n éléments (utilisé lors de l'analyse) en créant une liste de phrases
f. list_of_dicts (list_of_sentences, number) : transformation d'une liste de phrases en liste de dictionnaires d'occurrences. Le paramètre « number » correspond au modèle choisi (1 pour unigrammes, 2 pour bigrammes et 3 pour trigrammes)
g. Calcul_cos_lists : calcul le cos entre un dictionnaire de référence et une liste de référence (un cos par élément (dictionnaire) dans la liste) -> Renvoie une liste de cos.
    - Cette fonction est utilisée une fois par langue
h. compare_cos() : prend en paramètre les listes de cos et les compare une à une pour voir, pour chaque phrase, vers quelle langue elle se rapproche. Le paramètre "number" correspond au modèle choisi.
    - Incrémente le compteur des langues après chaque détection d'une langue
    - Détermine le maximum des compteurs des langues afin d'annoncer la langue du texte et l'accuracy (Max/nombre total de phrases)

3.2. Choix d'implémentation

[3]https://docs.google.com/spreadsheets/d/1UR3POVh4q2Qr6kkQJfjFEOTbYE55BPWbU_eLBULXDh0/edit#gid=0

- Librairies :

  import math : librairie utilisée pour le calcul de cosinus

  import sys : librairie utilisée pour l'interprétation des inputs dans l'invite de commande

  from collections import Counter : librairie utilisée pour compter les occurrences d'items dans un texte

  from nltk import bigrams, trigrams : librairie utilisée pour la détection des n-grammes. Ces fonctions renvoient une liste, nous avons donc dû le transformer en dictionnaire par la suite.
- Document gdocData :

  from __future__ import print_function

  from googleapiclient.discovery import build

  from google_auth_oauthlib.flow import InstalledAppFlow

  from google.auth.transport.requests import Request

  from google.oauth2.credentials import Credentials

  from google.oauth2 import service_account

Ces librairies sont utiles pour la relation entre notre programme et le Google sheet.

- Technologies :

  Langage : Python. Semblait approprié pour le traitement de données textuelles. Nous voulions par ailleurs nous entrainer afin de nous améliorer dans ce langage.

  Google sheet : Nous avions comme idée de départ d'enregistrer nos dictionnaires sur le Google sheet afin d'introduire un apprentissage automatique. Nous aurions donc ajouté notre nouveau texte détecté au dictionnaire déjà présent sur le Google sheet. Notre base de données augmenterait donc à chaque nouvelle utilisation du logiciel, ce qui aurait augmenté la précision de détection. Nous avons fait le choix de ne pas mettre cela en place car cela aurait compliqué notre analyse des modèles. En effet, il aurait fallu dissocier les dictionnaires d'unigrammes, de bigrammes et de trigrammes pour chaque langue. Cela ne semblait pas nécessaire avant d'avoir effectué l'analyse de performance des modèles. Nous aimerions cependant implémenter cela dans le futur.

Nous avons donc utilisé le Google sheet pour l'exportation des données des analyses du modèle et la comparaison des différents modèles selon la langue. Cela nous permettait d'avoir une vision claire des résultats et une représentation sous forme de tableau et de graphique sur la

même interface. La différence entre le Google sheet et un tableau Excel est la facilité que cela apportait pour travailler en groupe (partage et mise à jour en direct).

3.3. Problèmes rencontrés et solutions proposées

Compréhension du sujet : La difficulté principale a été dans la compréhension du sujet et des attentes. Nous avions initialement compris qu'il fallait uniquement déterminer la langue d'un texte et non analyser les différents modèles.
Nous avions trois modèles :

- Unigrammes.
- Unigrammes + bigrammes.
- Unigrammes + bigrammes + trigrammes.

Nous disposions de dictionnaires de référence que nous transformions en listes et nous comparions des listes au lieu de dictionnaires. Pour cela, nous devions créer des vecteurs creux et itérer sur ces listes. Cela résultait en de l'allocation de mémoire qui n'était pas nécessaire. Nous en avons discuté lors de la pré-soutenance et avons modifié cela depuis.

Nous n'avions pas non plus compris le principe d'accuracy et le fait qu'il fallait découper le texte en phrases et analyser ces différentes phrases.

Encodage : Une autre difficulté est l'encodage des fichiers textes pris en compte. Notre logiciel accepte uniquement l'UTF-8 (et tous les encodages associés). C'est pour cela que nous avons programmé le logiciel pour n'importer que les fichiers .txt car ce sont ceux en choix libre d'encodage.

La situation sanitaire : Le travail à distance n'a pas non plus été facile. Nous ne nous connaissions pas et nous ne nous sommes jamais vus. Nous n'avons jamais pu discuter du projet ensemble en face à face. Cependant, grâce à Discord, nous avons pu organiser des réunions hebdomadaires et discuter concrètement de notre projet, de son avancement, de s'organiser en taches, faire des croquis via le partage d'écran etc.
Pour le partage de documents, nous avons utilisé Github[4] afin de toujours avoir accès à la dernière version du code.

[4] https://github.com/theofal/ProjetTAL

4. Partie : Manuel utilisateur

4.1. Indications pratiques pour exécuter le programme

#Aide en ligne programmée

Projet TAL : Détection de la langue d'un texte

- **Présentation**

Ce projet est a été réalisé par Loubna Khennou et Théo Falgayrettes dans le cadre de notre L3 en informatique appliquée à la linguistique. Il s'agit d'un logiciel de détection de la langue d'un texte.

Nous savons que, par exemple, la lettre "w" apparait bien plus souvent dans les textes de langue anglaise que dans les textes français. La fréquence des caractères n'est donc pas la même selon les différentes langues.

Ce logiciel analyse un mot, un paragraphe ou un fichier texte et compare l'occurrence des caractères avec une base de données déjà présente. Ceci dans le but d'en déterminer la langue.

Les langues prises en charge actuellement sont le français, l'anglais, l'espagnol et l'italien.

- **Le statut du projet**

Ce projet est encore au stade de développement, avons quelques idées d'améliorations de notre logiciel :

-- Augmenter l'efficacité en augmentant la taille des mots analysés.

-- Mise en place d'un système d'apprentissage automatique augmentant la taille de notre corpus de référence lorsqu'un texte/une phrase a été reconnue comme appartenant à 100% à une langue (nécessitera une approbation manuelle au début).

-- Prise en charge de langues supplémentaires.

-- Création d'une interface graphique.

- **Les exigences concernant l'environnement de développement en vue de son intégration.**

L'utilisation de ce logiciel requiert l'installation préalable de Python : https://www.python.org/downloads/

- **Une instruction pour l'installation et l'utilisation.**

Une fois le fichier téléchargé, rendez-vous dans celui-ci via le terminal.
Il existe deux manières d'analyser un texte :
-- Analyse d'un fichier .txt : Python CodeProjet.py ~/monFichier.txt
-- Analyse d'un mot ou paragraphe en écriture libre: Python CodeProjet.py "mon paragraphe à analyser"

- **Les technologies utilisées**

Google sheet (afin d'analyser les données) : https://docs.google.com/spreadsheets/d/1UR3POVh4q2Qr6kkQJfjFEOTbYE55BPWbU_eLBULXDh0/edit?usp=sharing

Conclusion

Les résultats obtenus par ce programme montrent l'intérêt de la présence des modèles de langues dans le sujet de la détection de langue et sa relation avec la longueur du texte à analyse. C'est ainsi grâce à un corpus remarquable que nous réussirons à découvrir lequel des modèles de langue est plus performant. Nous devons donc partir d'un très large corpus et calculer la fréquence des caractères pour pouvoir estimer la probabilité de chaque n-gramme et par la suite détecter la langue du corpus car plus le corpus utilisé pour ces estimations est grand, plus on peut espérer obtenir des estimations de probabilités juste. En revanche, quelle que soit la taille du corpus d'apprentissage, bien que le modèle des n-grammes est bel est bien simple efficace, il demeure très limité.
Autrement dit, ce dernier ne permet pas de capter les dépendances entre les caractères et les symboles qui sont éloignés dans la séquence. D'ailleurs, lors du traitement d'un texte, il se peut qu'un n-gramme qui ne se trouve pas dans le modèle apparait. L'estimation de probabilité de ce n-gramme sera alors de 0 et le modèle ne sera pas donc aussi performant.
La conséquence de cette probabilité nulle est le fait d'attribuer une probabilité nulle à toutes la séquence de caractères contenant un n-gramme non rencontré dans le corpus.
Les modèles donc construits ne sauraient reconnaitre que les phrases dont les n-grammes sont tous apparus dans le corpus. C'est pour cette raison et afin de généraliser les modèles, nous devons assouplir cette attribution systématique de probabilité nulle aux séquences de caractères non rencontrés. Dans ce cadre, nous pourrons faire appel au lissage pour attribuer une

probabilité non-nulle. Or, est ce que cette procédure va permettre effectivement de doter les trois modèles n-grammes d'une plus grande capacité de généralisation ?