Adrien Vergne

RAPPORT

Programmation de Modèles Linguistiques (II)

Année universitaire 2024-2025 – L5SOPROG / L3 Sciences du Langage

1. **CONTEXTE ET OBJECTIFS**

Ces travaux pratiques (TD) portent sur deux axes principaux :

Analyse de langue en caractères (tri-grammes) Améliorer un système d'identification de langue déjà existant (baseline en mots) en utilisant des tri-grammes de caractères.

- Évaluer le nouveau modèle et fournir un diagnostic de langue (avec score de confiance).

- Comparer les langues « proches » et représenter graphiquement ces proximités.

- Reconnaissance d'entités nommées au format IOB Utiliser un jeu de données déjà annoté (fichiers CSV) et des textes à annoter automatiquement (avec spaCy).

- Convertir les annotations au format IOB, puis aligner et évaluer (précision, rappel, F1-score).

- Sauvegarder les résultats et mesurer le temps de traitement.

- L'ensemble des scripts a été développé sous Spyder, conformément aux consignes, et factorisé pour respecter les bonnes pratiques de la programmation (structures de données pertinentes, modularité, etc.). Les modules et bibliothèques clés utilisés incluent notamment :

- json, csv, os, glob, time

- numpy, pandas

- spacy

- sklearn (CountVectorizer, TfIdfVectorizer, et diverses métriques)

- matplotlib, seaborn (visualisation)

2. **IDENTIFICATION DE LANGUE À PARTIR DE TRI-GRAMMES DE CARACTÈRES**

2.1. Rappels et méthodologie

Dans le cadre du TD4 du semestre précédent, une baseline d'identification de langue fonctionnait à partir d'un vocabulaire de mots. L'amélioration proposée consiste à :

1. Représenter les textes via des tri-grammes de caractères plutôt qu'en mots.
2. Mesurer la similarité (Cosine Similarity) entre les documents de test et les centroïdes calculés pour chaque langue.
3. Faire la prédiction de la langue qui présente la plus grande similarité, tout en indiquant :
   - Un score de confiance (pourcentage).
   - Les autres langues potentiellement proches (top_k).

Données :

- Un corpus multilingue organisé par langue, réparti en sous-dossiers appr/ (apprentissage) et test/ (évaluation).

- Le programme lit l'ensemble des fichiers, regroupe les textes par langue, et construit un vecteur moyen (centroïde) pour chaque langue.

- Principales étapes du script :

- Ouverture et lecture des données (fonctions open_entrainementDataBase, glob.glob).

- Vectorisation avec CountVectorizer (paramétré sur des caractères, ngram_range=(3,3)).

- Construction des centroïdes : calcul de la moyenne des vecteurs représentant chaque document pour une même langue.

Prédiction (PredictLangues) :

- Vectoriser un texte de test.

- Calculer la similarité cosinus avec le centroïde de chaque langue.

- Choisir la langue à la similarité la plus élevée et estimer un score de confiance.

Évaluation (PyProgWork et CalculPrediction) :

- Parcourir tous les fichiers de test, comparer la prédiction (pred) avec la référence (ref).

- Générer une matrice de confusion et un rapport de classification (précision, rappel, F1-score), puis calculer l'accuracy.

- Visualisation des proximités entre langues (PlotLangDist) :

- Construire la matrice de similarités (cosinus) entre chaque centroïde.

- Convertir en distance (1 - similarité) et afficher sous forme de carte de chaleur (seaborn/matplotlib).

2.2. Résultats obtenus

- L'exemple de matrice de confusion présenté montre qu'en moyenne, la prédiction est correcte dans 98 % des cas.

Extrait des résultats :

- Matrice de confusion (lignes=référence, colonnes=prédiction) : cs, da, de, el, en, es, etc.

- La langue en (Anglais) reste correctement identifiée, avec parfois un léger décalage (confusions minimes).

- Le rapport de classification indique une exactitude globale d'environ 0,98.

- Certaines langues sont identifiées à 100 %, d'autres à 98-99 %.

- Ce score de 98 % valide l'idée que la prise en compte de tri-grammes de caractères est efficace pour l'identification de la langue, y compris lorsqu'il s'agit de langues proches.

2.3. Représentation graphique des langues proches

- La fonction PlotLangDist produit une heatmap (carte de chaleur) des distances entre les centroïdes. Les langues proches (par exemple, fr, it, es, pt) apparaissent dans des zones où la distance est plus faible. Les langues plus éloignées montrent une distance plus importante.

3. **RECONNAISSANCE D'ENTITÉS NOMMÉES (NER) AU FORMAT IOB**

3.1. Données et format attendu

- Jeu de données annoté (CSV) avec les colonnes Token, LOC, PER, ORG, MISC. Les cellules contiennent B, I ou O.

- Textes bruts à annoter automatiquement.

- Le format IOB (Inside, Outside, Beginning) est utilisé pour les données gold (CSV) et pour les - annotations automatiques (spaCy).

3.2. Méthode de travail

1. Lecture et conversion des CSV :

   - Lire les tokens et marquages d'entités.

   - Construire des tuples (token, iob_tag, ent_type) du type ("San", "B", "LOC").

   - Sauvegarder au format .bio (fichier texte).

2. Annotation automatique avec spaCy :

   - Charger un modèle de langue (ex. en_core_web_sm).

   - Parcourir chaque token et récupérer token.ent_iob_, token.ent_type_.

- Sauvegarder également au format .bio.

3. Alignement et évaluation :

- Parcourir les fichiers gold et auto ayant le même nom de base.

- Aligner naïvement les tokens (index par index).

- Calculer TP, FP, FN afin d'établir la précision, le rappel, et le F1.

4. Résultats :

- Affichage ou export des scores par fichier.

- Mesure du temps de traitement total.

3.3. Résultats observés

Les étapes d'exécution s'affichent en console :

- Conversion des CSV (gold) au format IOB.

- Annotation automatique des textes (spaCy).

- Alignement et calcul des métriques (précision, rappel, F1).

Dans l'exemple, environ 45 secondes de temps de traitement total. Les scores varient selon la cohérence entre l'annotation gold et spaCy. D'éventuelles divergences apparaissent lorsque spaCy segmente ou étiquette différemment des entités (par exemple, GPE vs LOC).

4. **CONCLUSION ET PERSPECTIVES**

Identification de langue :

- L'approche par tri-grammes de caractères et similarité cosinus entre centroïdes atteint 98 % de réussite sur le corpus de test.

- Les langues proches restent légèrement plus difficiles à différencier, mais le taux de confusion demeure faible.

Reconnaissance d'entités nommées :

- L'usage du format IOB facilite la comparaison entre annotations automatiques et manuelles (gold).

- L'alignement naïf permet de calculer précision, rappel et F1.

En définitive, ces TD ont permis de :

- Manipuler des structures de données et bibliothèques standard (json, csv, collections).

- Appliquer des bibliothèques spécialisées (scikit-learn, spaCy) à des problématiques de TAL.

- Évaluer et visualiser les performances des modèles, tout en respectant de bonnes pratiques de programmation.

Des pistes d'amélioration incluent l'optimisation des n-grammes (TF-IDF, prise en compte de la ponctuation), l'entraînement de modèles NER plus spécialisés, ou l'exploration de méthodes de deep learning pour la classification de langue et la détection d'entités nommées.

5. **LIVRABLES**

Conformément aux consignes, le dépôt comprend :

Un script Python unique (.py) pour chaque exercice :

1. Exo1 : Identification de langue.
2. Exo2 : Reconnaissance d'entités nommées en IOB avec spaCy.

Les fichiers de sortie (.bio, .json, etc.) dans un répertoire de résultats.

Un document PDF (ce rapport) résumant la méthodologie, les résultats et quelques points de conclusion.

La date limite et les modalités de remise sont disponibles sur la plateforme Moodle.