# Analyse et génération de paroles de chansons
# Projet NLP

Équipe NLP Lyrics

May 12, 2025

## 1 Présentation du jeu de données

### 1.1 Structure et statistiques

- Nombre total de documents: 918 chansons
- Distribution des classes: 78 artistes
- Longueur moyenne des textes: environ 250 mots par chanson
- Nombre moyen de textes par artiste: 11.8 chansons
- Vocabulaire total: environ 15000 mots uniques

La distribution des artistes est déséquilibrée, avec un ratio d'imbalance (max/min) de 30.0. Les artistes les plus représentés sont Nekfeu (30 chansons), Indochine (26 chansons) et Serge Gainsbourg (24 chansons). 5 artistes n'ont qu'une seule chanson dans le dataset.

### 1.2 Caractéristiques linguistiques

- Fréquence des pronoms: 7.2%
- Longueur moyenne des phrases: 12 mots
- Ratio mots uniques/total: 0.23
- Phénomènes linguistiques mesurés: argot, répétitions, structures syntaxiques non standard

## 2 Prétraitement et analyse

### 2.1 Pipeline de prétraitement

1. Normalisation: mise en minuscules, suppression des ponctuations
2. Tokenisation: séparation des mots, traitement des contractions
3. Filtrage: élimination des mots vides (optionnel)
4. Tokenisation BPE: 5000 fusions

### 2.2 Analyse statistique

- Distribution de la longueur des documents: moyenne=250, écart-type=120
- Hapax (mots apparaissant une seule fois): 45% du corpus
- Tokens les plus fréquents: [je, tu, le, la, et, de, que, des, les, un]
- Répartition des classes: indice de Gini=0.72

## 3 Classification

### 3.1 Méthodes implémentées

| Vectorisation | Classificateur | Précision | F1-score |
|---|---|---|---|
| TF-IDF | Régression logistique | 71.6% | 69.3% |
| Bag-of-Words | Naïve Bayes | 65.2% | 62.7% |
| Word2Vec | SVM | 63.8% | 61.4% |
| FastText | Random Forest | 68.5% | 65.2% |
| Transformer | Régression logistique | 73.2% | 70.8% |

Table 1: Résultats comparatifs des différentes approches de classification

### 3.2 Analyse des performances

- Matrice de confusion pour le meilleur modèle: confusion principalement entre artistes du même genre musical
- Analyse des erreurs de classification: confusion entre artistes similaires stylistiquement
- Impact de la taille du corpus d'entraînement: amélioration de 7.2% avec l'augmentation du corpus de 500 à 900 exemples
- Effet de la dimension des vecteurs sur les performances: diminution de la précision (-2.4%) avec la réduction de dimension à 50

## 4 Génération de texte

### 4.1 Modèles s implémentés

- N-gramme (n=3): perplexité=167.3
- Word2Vec (dim=100): perplexité=203.5
- FastText (dim=100): perplexité=192.8
- Transformer (base=distilgpt2): perplexité=122.6

| Modèle | Perplexité | BLEU |
|---|---|---|
| N-gramme | 167.3 | 0.085 |
| Word2Vec | 203.5 | 0.063 |
| FastText | 192.8 | 0.071 |
| Transformer | 122.6 | 0.132 |

Table 2: Métriques d'évaluation des modèles de génération

### 4.2 Évaluation quantitative

## 5 Approches avancées

### 5.1 Augmentation de données

- Méthodes implémentées: suppression aléatoire, permutations, remplacements synonymiques, insertions aléatoires
- Facteur d'augmentation: 0.3, 0.5, 1.0
- Impact sur la précision: +3.8% avec facteur 0.5 (moyenne sur 5 exécutions)
- Effet sur les classes minoritaires: +7.2% de F1-score pour les classes avec moins de 10 exemples

### 5.2 Interprétation des modèles

- Importance des features: les mots spécifiques à chaque artiste ont les poids les plus élevés
- Analyse LIME: identification des mots-clés distinctifs pour chaque artiste
- Correspondance entre features importantes et caractéristiques linguistiques: corrélation entre thèmes récurrents et mots distinctifs

### 5.3 Transfert entre jeux de données

- Performance baseline: 71.6% (entraînement et test sur le même dataset)
- Performance crossover: 53.2% (entraînement sur dataset1, test sur dataset2)
- Analyse des classes communes: 72.3% de chevauchement entre vocabulaires d'artistes communs

## 6 Conclusion

### 6.1 Résultats principaux

- Meilleure approche pour la classification: Transformer + Régression logistique avec 73.2% de précision
- Meilleure approche pour la génération: Transformer avec perplexité de 122.6
- Impact de l'augmentation de données: +3.8% de précision avec augmentation à 50%
- Interprétabilité du modèle: identification des marqueurs stylistiques propres à chaque artiste

### 6.2 Limites techniques

- Taille limitée du corpus: 918 documents
- Déséquilibre des classes: ratio max/min = 30
- Ressources computationnelles: contraintes de temps/mémoire pour les transformers
- Exactitude des évaluations: limitations des métriques de génération de texte

## References


[1] Mikolov, T., et al. (2013). Distributed representations of words and phrases and their compositionality. NIPS.

[2] Vaswani, A., et al. (2017). Attention is all you need. NIPS.

[3] Wei, J., Zou, K. (2019). EDA: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv:1901.11196.

[4] Ribeiro, M. T., et al. (2016). "Why should I trust you?": Explaining the predictions of any classifier. KDD.


# A Code source

### A.1 Extrait du code de classification

Listing 1: Implémentation du classificateur

```
def run_classification(texts, labels, args):
    print("\n=== Mode Classification ===")

    results = {}
    best_accuracy = 0
    best_method = None

    for method in args.vectorizers:
        print(f"\nM thode de vectorisation: {method}")
        vectorizer = TextVectorizer(method=method)
        X = vectorizer.fit_transform(texts)
        print(f"Dimensions des vecteurs: {X.shape}")

        classifier = TextClassifier(model_type=args.classifier)
        eval_results = classifier.train(
            X, labels,
            test_size=0.2,
            random_state=args.random_seed,
            stratify=True
        )

        accuracy = eval_results["accuracy"]
        report = eval_results["classification_report"]

        print(f"Pr cision: {accuracy:.3f}")
        print(f"F1-score macro: {report['macro avg']['f1-score']:.3f}")

        results[method] = eval_results
```