Rapport de projet de fin d'année

Classification d'hallucination de modèles de langage

Encadrée par Dr. Pierre De Loor & Benjamin Vendeville

RÉALISÉ PAR Tarrass Rajaa

Établissement d'enseignement: Ecole Nationale d'Ingénieurs de Brest (ENIB)
Année universitaire 2024-2025

Contents

1 Introduction

Avec l'essor des modèles de langage de grande envergure (LLM) tels que GPT et BERT, les applications d'intelligence artificielle dans le traitement du langage naturel (NLP) ont considérablement progressé. Parmi ces applications, la simplification automatique de texte est devenue un domaine clé, notamment pour rendre les textes scientifiques accessibles à un public non-expert. Cependant, ces modèles génèrent souvent des résultats erronés, qualifiés d'hallucinations, qui peuvent inclure des informations hors contexte, des pertes d'informations cruciales ou des erreurs syntaxiques.

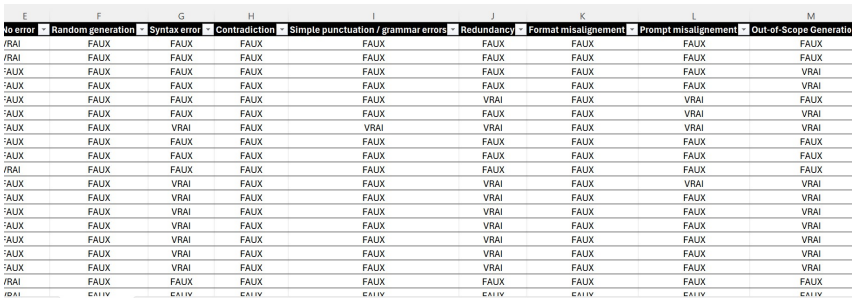
Ce projet s'inscrit dans le cadre d'une étude visant à analyser et à classifier ces hallucinations. Plus précisément, l'objectif est de développer un pipeline capable de détecter et de classifier certains types d'erreurs spécifiques, tels que :

- **No error** : lorsque la génération est correcte et répond pleinement aux attentes.
- **Random generation** : lorsque le contenu généré est aléatoire et sans rapport avec la source.
- **Syntax error** : lorsque des erreurs grammaticales ou structurelles apparaissent dans le texte généré.
- **Contradiction** : lorsque le contenu généré contredit les informations de la source.
- **Simple punctuation / grammar errors** : lorsque des erreurs mineures de ponctuation ou de grammaire sont présentes.
- **Redundancy** : lorsque des informations inutiles ou répétitives apparaissent.
- **Format misalignment** : lorsque la structure ou le format attendu ne sont pas respectés.
- **Prompt misalignment** : lorsque la génération ne correspond pas aux instructions ou au contexte initial.
- **Out-of-Scope Generation** : lorsque des informations générées sont hors contexte ou non pertinentes.
- **Topic shift** : lorsque le contenu s'éloigne complètement du sujet initial.
- **Oversimplification of Logical Arguments** : lorsque des arguments logiques sont simplifiés de manière excessive, compromettant leur sens.
- **Overgeneralization** : lorsque des informations sont généralisées de manière incorrecte.
- **Loss of Informative Content** : lorsque des informations essentielles du texte source sont omises.
- **Factuality hallucination** : lorsque des faits incorrects ou inexistants sont générés.
- **Faithfulness hallucination** : lorsque le texte généré ne reste pas fidèle à l'information source.

Pour répondre à cet objectif, des techniques d'apprentissage automatique ont été explorées. Dans ce rapport, nous présenterons en détail les étapes principales de ce projet.

2 Extraction des donnees

jai débuté ce projet en utilisant les données fournies par l'équipe responsable des annotations manuelles, composée notamment de Monsieur Benjamin, Pierre et Amélie. Bien que le dataset initial soit de taille réduite, il a servi de base pour constituer un ensemble de données complet et adapté à nos analyses. Par ailleurs, jai opté pour une structuration des données au format CSV, afin de faciliter leur exploitation dans les étapes ultérieures.



E	F	G	H	I	J	K	L	M
No error	Random generation	Syntax error	Contradiction	Simple punctuation / grammar errors	Redundancy	Format misalignement	Prompt misalignement	Out-of-Scope Generatio
/RAI	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
/RAI	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
'AUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	VRAI
'AUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	VRAI
'AUX	FAUX	FAUX	FAUX	FAUX	VRAI	FAUX	VRAI	FAUX
'AUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	VRAI	VRAI
'AUX	FAUX	VRAI	FAUX	VRAI	VRAI	FAUX	VRAI	VRAI
'AUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
'AUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
/RAI	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	VRAI	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
'AUX	FAUX	VRAI	FAUX	FAUX	VRAI	FAUX	FAUX	VRAI
/RAI	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX	FAUX

Figure 1: dataset

3 EDA - analyse exploratoire des données

4 Analyse Exploratoire des Données (EDA)

Pour débuter notre analyse, nous avons examiné les valeurs manquantes dans le dataset. Nous avons constaté que la colonne *commentaires* des annotateurs contenait 90% de valeurs manquantes. Étant donné que cette colonne n'est pas pertinente pour notre projet, elle a été supprimée. Les résultats généraux du dataset après cette étape sont résumés dans le tableau ci-dessous :

Dataset	**Rows**	**Columns**	**%Duplicate**	**object_dtype**	**bool_dtype**
merged dataframe	330	20	0.0	6	14

Table 1: Résumé des propriétés du dataset après traitement des valeurs manquantes.

Ensuite, nous avons examiné les types d'erreurs présentes dans le dataset et avons calculé le pourcentage de chaque type. Le tableau ci-dessous présente les résultats, triés par ordre décroissant des occurrences de la classe *True* :

Type d'erreur	**True (%)**	**False (%)**
Loss of Informative Content	20.606061	79.393939
Out-of-Scope Generation	13.939394	86.060606
Overgeneralization	12.727273	87.272727
Syntax error	10.000000	90.000000
Simple punctuation / grammar errors	9.090909	90.909091
Faithfulness hallucination	8.181818	91.818182
Oversimplification of Logical Arguments	7.575758	92.424242
Redundancy	7.272727	92.727273
Topic shift	5.757576	94.242424
Format misalignement	5.151515	94.848485
Random generation	4.545455	95.454545
Prompt misalignement	1.212121	98.787879
Factuality hallucination	0.909091	99.090909
Contradiction	0.000000	100.000000

Table 2: Distribution des types d'erreurs dans le dataset, en pourcentage.

En examinant la répartition des types d'erreurs dans le dataset, j'ai constaté que la classification serait déséquilibrée, car les pourcentages de la classe *True* sont très faibles pour la plupart des types d'erreurs. Certains types ne contiennent même aucune occurrence de *True*. Par conséquent, j'ai décidé de limiter l'apprentissage à quelques types d'erreurs pour ce projet initial.

Pour commencer, j'ai choisi de me concentrer sur deux types d'erreurs spécifiques :

- **Loss of Informative Content** : avec un pourcentage de *True* de 20.61 %.
- **Out-of-Scope Generation** : avec un pourcentage de *True* de 13.94 %.

Ces deux types présentent les pourcentages de *True* les plus élevés dans le dataset, ce qui permet de les classifier de manière plus réaliste. Cependant, avec un futur dataset mieux équilibré et de plus grande taille, il sera envisageable d'étendre cette approche à la classification de tous les types d'erreurs. Pour l'instant, ces choix permettent de poser une base solide pour entamer le projet tout en tenant compte des limitations actuelles des données.

5 Nettoyage et Prétraitement des Textes

Le processus de prétraitement des *source sentence* et *simplified sentence* a englobé plusieurs étapes essentielles pour garantir des données prêtes à l'emploi avant la génération des embeddings. Ces étapes incluent :

- **Suppression de la ponctuation** : élimination des caractères de ponctuation inutiles pour éviter les interférences dans l'analyse.
- **Conversion en minuscules** : homogénéisation des textes pour réduire les variations dues à la casse.
- **Suppression des stopwords** : élimination des mots courants et peu informatifs tels que "the", "is", "and", etc.
- **Lemmatisation** : réduction des mots à leur forme canonique pour un traitement sémantique cohérent.

Ces étapes sont cruciales pour nettoyer les données textuelles et garantir une représentation vectorielle optimale lors de la phase d'embedding. Elles permettent également de réduire le bruit et d'améliorer la qualité des données utilisées pour l'apprentissage.

6 Génération des Embeddings

Dans le cadre de ce projet, deux approches distinctes ont été explorées pour générer les embeddings textuels des *source sentence* et *simplified sentence*. Ces embeddings sont essentiels pour capturer les relations sémantiques entre les phrases et permettre la classification des types d'erreurs.

Figure 2: HF

6.1 Sentence Embedding avec Sentence-BERT

La première méthode repose sur l'utilisation du modèle **Sentence-BERT (all-MiniLM-L6-v2)**, qui est optimisé pour produire des embeddings vectoriels représentant le sens global des phrases. Voici les étapes principales de cette approche :

- **Objectif** : Générer des représentations vectorielles compactes pour des phrases, optimisées pour des tâches de similarité sémantique et de classification.
- **Avantages** :

- Léger et rapide, adapté à des tâches où les ressources de calcul sont limitées.
- Génère des embeddings de haute qualité, bien adaptés pour des tâches simples de comparaison sémantique.

- **Limites** :
  - Moins performant pour capturer des relations complexes entre des textes comparés à des modèles plus récents et spécialisés.

6.2 Embedding avec le Modèle BGE-M3

La deuxième méthode utilise le modèle **BAAI/bge-m3 (Base General Embedding - M3)**, disponible sur Hugging Face. Ce modèle, conçu spécifiquement pour produire des embeddings polyvalents et riches en sémantique, a démontré de meilleures performances dans le contexte de ce projet. Voici ses principales caractéristiques :

Description du modèle BGE-M3 :

- **Objectif** : Générer des embeddings vectoriels riches et significatifs, adaptés à des tâches telles que la classification, la recherche sémantique et la correspondance de documents.
- **Architecture** : Basé sur un modèle transformer, probablement dérivé de BERT, avec des ajustements pour maximiser la qualité des embeddings.
- **Points forts** :
  - Compression efficace : Les vecteurs générés sont denses et adaptés à des comparaisons dans un espace latent.
  - Polyvalence : Convient à une large gamme de langues et de domaines grâce à un entraînement sur des corpus diversifiés.
  - Optimisation GPU (FP16) : Compatible avec les calculs en demi-précision, permettant d'accélérer les calculs sur des machines équipées de GPU.

7 Feature Engineering

Dans cette étape, nous avons enrichi notre dataset en ajoutant une nouvelle colonne appelée *similarity*, représentant la similarité cosinus entre les espaces latents des *source sentence* et *simplified sentence*. Cette colonne constitue une caractéristique clé pour la classification, permettant de capturer la relation sémantique entre les deux textes.

7.1 Calcul de la Similarité Cosinus

Le calcul de la similarité cosinus a été réalisé à l'aide de la fonction `cosine_similarity` de la bibliothèque `sklearn`. Cette mesure est définie comme suit :

$$\text{similarity} = \frac{\vec{A} \cdot \vec{B}}{\|\vec{A}\|\|\vec{B}\|}$$

où $\vec{A}$ et $\vec{B}$ sont respectivement les vecteurs latents des *source sentence* et *simplified sentence*.

7.2 Rôle de la Colonne Similarity

La colonne *similarity* est essentielle dans notre approche, car elle fournit une mesure quantitative de la proximité sémantique entre la phrase source et la phrase simplifiée. Une similarité élevée peut indiquer que le modèle a préservé le sens initial de la phrase, tandis qu'une similarité faible peut signaler une perte d'information ou un contenu hors contexte.

Cette caractéristique s'avère particulièrement utile pour classifier les types d'erreurs suivants :

- **Loss of Informative Content** : lorsque des informations essentielles sont omises.
- **Out-of-Scope Generation** : lorsque le contenu généré est hors du contexte de la source.

En ajoutant cette colonne, nous avons significativement enrichi le dataset et amélioré la qualité des données utilisées pour l'entraînement des modèles de classification.

8 Modélisation

Avant de débuter le processus d'apprentissage, nous avons choisi d'effectuer deux classifications distinctes pour les types d'erreurs suivants :

- **Loss of Informative Content** : identification des cas où des informations essentielles sont omises.
- **Out-of-Scope Generation** : détection des cas où le contenu généré est hors du contexte attendu.

Plutôt que de combiner les deux classifications dans un même modèle d'apprentissage automatique, nous avons opté pour un apprentissage supervisé indépendant pour chaque type d'erreur. Cette approche nous a permis de mieux adapter les modèles aux spécificités de chaque tâche.

Dans ce rapport, nous allons présenter uniquement les résultats obtenus pour la classification *Loss of Informative Content*. Bien que les deux types d'erreurs aient été analysés dans les codes disponibles en annexe, nous avons choisi de ne détailler que ce cas afin de ne pas alourdir le contenu de ce document.

8.1 Division du Dataset : Split Train/Test

Pour préparer les données à l'entraînement, nous avons divisé le dataset en deux ensembles:

- 80% des données pour l'entraînement.
- 20% des données pour le test.

8.2 SMOTE : Oversampling de la Classe Minoritaire

Étant donné le caractère déséquilibré du dataset, où la classe *True* est significativement sous-représentée par rapport à la classe *False*, nous avons appliqué l'algorithme **SMOTE (Synthetic Minority Oversampling Technique)**. Cet oversampling génère des exemples synthétiques pour la classe minoritaire, permettant ainsi de :

- Rééquilibrer les classes dans le dataset d'entraînement.
- Éviter que les modèles ne biaisent leurs prédictions en faveur de la classe majoritaire.

Cette étape est cruciale pour améliorer les performances du modèle, en particulier pour le rappel et le F1-score de la classe minoritaire.

8.3 Logistic Regression

Nous avons utilisé la régression logistique comme premier modèle d'apprentissage supervisé. Ce modèle linéaire est simple mais efficace pour les tâches de classification binaire. Voici les résultats obtenus après entraînement et évaluation :

- **Matrice de confusion :**

```
[[46 10]
 [ 3  7]]
```

- **Rapport de classification :**

	precision	recall	f1-score	support
False	0.94	0.82	0.88	56
True	0.41	0.70	0.52	10
accuracy			0.80	66
macro avg	0.68	0.76	0.70	66
weighted avg	0.86	0.80	0.82	66

- **Balanced Accuracy :**

  La Balanced Accuracyest une métrique utilisée pour évaluer les performances d'un modèle de classification sur des données déséquilibrées. Elle correspond à la moyenne du taux de rappel (*recall*) pour chaque classe. Cette métrique permet de prendre en compte les déséquilibres entre les classes en donnant un poids égal à chaque classe, indépendamment de leur fréquence dans le dataset.

  La formule de la *Balanced Accuracy* est donnée par :

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} + \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}\right)$$

  Dans le cadre de ce projet, la *Balanced Accuracy* obtenue pour le modèle est de :

$$\text{Balanced Accuracy} = 0.761$$

Les résultats montrent que le modèle est performant pour prédire la classe majoritaire *False*, mais prédit moyennement la classe minoritaire *True*. Cependant, l'oversampling via SMOTE a permis d'améliorer le rappel et le F1-score de la classe minoritaire.

8.4 SVM : Support Vector Machines

Nous avons également exploré les machines à vecteurs de support (SVM) comme modèle d'apprentissage supervisé. Ce modèle est particulièrement efficace pour les données déséquilibrées lorsqu'il est associé à un hyperparamètre de pondération adapté. Les résultats obtenus sont les suivants :

- **Matrice de confusion :**

```
[[46 10]
 [ 3  7]]
```

- **Rapport de classification :**

	precision	recall	f1-score	support
False	0.94	0.82	0.88	56
True	0.41	0.70	0.52	10
accuracy			0.80	66
macro avg	0.68	0.76	0.70	66
weighted avg	0.86	0.80	0.82	66

- **Balanced Accuracy :** 0.761

Comme pour la régression logistique, le modèle SVM affiche une bonne précision globale, mais des performances limitées pour la classe minoritaire.

8.5 Grid Search et Optimisation des Hyperparamètres

Pour optimiser les performances des modèles, nous avons appliqué une recherche d'hyper paramètres Grid Search en utilisant une validation croisée. Cette étape a permis d'identifier les combinaisons optimales pour maximiser les métriques, en particulier le F1-score. Grâce à cette optimisation, nous avons obtenu un F1-score maximal de 88%, démontrant l'efficacité de l'ajustement des paramètres sur les performances des modèles.

8.6 Performances avec le Modèle BGE-M3

En utilisant le modèle d'embedding **BAAI/bge-m3 (Base General Embedding - M3)**, nous avons obtenu des résultats significativement améliorés, comparés à ceux obtenus avec Sentence-BERT (*all-MiniLM-L6-v2*). Les résultats des deux algorithmes testés, *Régression Logistique* et *SVM*, sont présentés ci-dessous.

Régression Logistique

- **Matrice de confusion :**

```
[[50  6]
 [ 2  8]]
```

- **Rapport de classification :**

	precision	recall	f1-score	support
False	0.96	0.89	0.93	56
True	0.57	0.80	0.67	10
accuracy			0.88	66
macro avg	0.77	0.85	0.80	66
weighted avg	0.90	0.88	0.89	66

- **Balanced Accuracy :** 0.846

SVM

- **Matrice de confusion :**

```
[[50  6]
 [ 4  6]]
```

- **Rapport de classification :**

	precision	recall	f1-score	support
False	0.93	0.89	0.91	56
True	0.50	0.60	0.55	10
accuracy			0.85	66
macro avg	0.71	0.75	0.73	66
weighted avg	0.86	0.85	0.85	66

- **Balanced Accuracy :** 0.746

8.7 Pourquoi le Modèle BGE-M3 a Donné de Meilleurs Résultats ?

Le modèle **BGE-M3** a surpassé Sentence-BERT dans ce projet pour plusieurs raisons :

1. **Qualité des embeddings** : Le modèle capture des relations sémantiques complexes avec une précision supérieure, en particulier pour des textes scientifiques contenant des nuances fines. Cela améliore considérablement la qualité des représentations vectorielles.

2. **Alignement sémantique** : Le modèle est particulièrement efficace pour comparer des phrases similaires tout en mettant en évidence des différences significatives dans les informations qu'elles contiennent. Cela est essentiel pour identifier des erreurs comme *Loss of Informative Content* et *Out-of-Scope Generation*.

3. **Adaptation aux tâches** : Entraîné sur des objectifs spécifiques, tels que la recherche sémantique et la classification, le modèle répond parfaitement aux besoins de ce projet. Son optimisation pour des tâches NLP variées lui confère une polyvalence qui dépasse celle des modèles généralistes.

8.8 Résumé

L'utilisation de BGE-M3 a permis d'améliorer les métriques de performance, notamment la *Balanced Accuracy* et le *F1-score*, pour les deux algorithmes testés. Ces résultats démontrent que le choix du modèle d'embedding joue un rôle crucial dans la réussite des tâches de classification, en particulier dans un contexte de déséquilibre de classes et de complexité sémantique.

9 Travaux à Venir

Les étapes futures de ce projet visent à élargir les capacités de classification et à proposer une solution interactive pour les utilisateurs. Ces travaux incluront deux axes principaux

9.1 Classification de Tous les Types d'Erreurs

Dans les prochaines étapes, nous prévoyons d'étendre la classification pour inclure l'ensemble des types d'erreurs présents dans le dataset. Cela nécessitera :

- L'augmentation du dataset pour résoudre les problèmes de déséquilibre de classes identifiés dans ce projet.

- L'optimisation des modèles existants ou l'exploration de nouveaux modèles adaptés aux données enrichies.

- La comparaison systématique des performances sur tous les types d'erreurs pour identifier les forces et les limites des approches actuelles.

9.2 Implémentation d'une Interface Utilisateur

Un autre objectif clé sera de concevoir et d'implémenter une interface utilisateur intuitive, permettant aux utilisateurs finaux de tester le pipeline de classification. Cette interface permettra :

- De saisir une *source sentence* et une *simplified sentence* générée par un modèle de langage (LLM).
- De classifier automatiquement les types d'erreurs présents dans la simplification générée.
- De fournir un retour immédiat sur la qualité et les problèmes identifiés dans la simplification.

Cette interface pourra être déployée sous forme d'application web interactive, intégrant les modèles de classification développés dans ce projet. L'objectif final est de fournir un outil pratique et accessible pour analyser les simplifications générées par les LLM, tout en renforçant leur fiabilité et leur qualité.

9.3 Résumé

Ces travaux futurs contribueront non seulement à améliorer les performances des modèles sur tous les types d'erreurs, mais également à offrir une solution conviviale et fonctionnelle pour des cas d'utilisation réels. Cela positionnera le projet comme une avancée significative dans l'analyse et la correction des sorties des modèles de langage de grande envergure.