REVUE DE LITTERATURE

# Classification d'Intentions Multi-Categorie

*pour la Detection de Requetes de Trajets*

# Table des Matieres

1. 1. Introduction
2. 2. Comprehension Approfondie de la Tache

* 2.1. Contexte et Objectifs
* 2.2. Definitions des Classes
* 2.3. Defis et Contraintes

1. 3. Architecture des Transformers

* 3.1. Principe Fondamental
* 3.2. Mecanisme d'Attention
* 3.3. CamemBERT pour le Francais

1. 4. Methodologie d'Implementation

* 4.1. Fine-Tuning pour Classification
* 4.2. Named Entity Recognition
* 4.3. Post-Processing

1. 5. Implementation Google Colab

* 5.1. Architecture Complete
* 5.2. Exemple Input-Output
* 5.3. Pipeline d'Inference

1. 6. Resultats et Discussion
2. 7. References

# 1. Introduction

La classification d'intentions est une tache fondamentale en traitement automatique du langage naturel (NLP) qui consiste a determiner l'intention sous-jacente d'un enonce utilisateur. Dans le contexte des systemes de dialogue et des assistants virtuels, cette capacite permet d'acheminer efficacement les requetes vers les modules de traitement appropries.

Ce projet s'inscrit dans le cadre d'un systeme plus large de recherche d'itineraires optimaux. L'objectif principal est de developper un classifieur capable de distinguer les requetes de trajets purs des autres types de demandes liees au voyage (horaires, tarifs, disponibilites) afin d'orienter l'utilisateur vers le service adequat.

Cette revue de litterature presente une analyse approfondie de la tache confiee, des architectures Transformers utilisees, notamment CamemBERT optimise pour le francais, et une implementation complete sous Google Colab avec des exemples concrets d'entree-sortie.

# 2. Comprehension Approfondie de la Tache

## 2.1. Contexte et Objectifs

Le systeme de classification d'intentions developpe repond a un besoin specifique dans le domaine de la mobilite ferroviaire. Il fait partie integrante d'un pipeline de traitement des requetes utilisateur pour un service de recherche d'itineraires.

**Objectifs principaux :**

* Identifier les requetes de trajet pur (point A vers point B)
* Distinguer les demandes d'information voyage (horaires, prix, billets)
* Detecter les requetes en langues etrangeres
* Filtrer les textes incomprehensibles ou hors contexte

## 2.2. Definitions des Classes

La tache de classification repose sur quatre categories distinctes, chacune correspondant a un type specifique de requete utilisateur :

**Classe TRIP (Trajet Pur)**

Definition : Requetes exprimant uniquement une demande de trajet d'un point de depart vers une destination, sans mention explicite de trains, billets, horaires ou tarifs.

*Exemples caracteristiques :*

* "De Paris a Lyon"
* "Je vais de Bordeaux a Marseille"
* "Strasbourg vers Metz"
* "J'aimerais aller de Nice a Cannes"
* "Bassila - Paris"

Cette classe necessite egalement l'extraction des entites geographiques (ville de depart et destination) via un module de Named Entity Recognition (NER).

**Classe NOT\_TRIP (Requete Voyage Non-Trajet)**

Definition : Requetes liees au domaine ferroviaire mais ne demandant pas directement un calcul d'itineraire. Inclut les demandes d'horaires, de billets, de disponibilites, de tarifs et de reservations.

*Exemples caracteristiques :*

* "Quels sont les horaires pour Paris Lyon ?"
* "Un billet pour Bordeaux Marseille"
* "Y a-t-il des trains disponibles de Nice a Cannes ?"
* "Quel est le prix du trajet Paris Lyon ?"
* "Je voudrais reserver une place pour demain"

**Classe NOT\_FRENCH (Langue Etrangere)**

Definition : Requetes formulees dans une langue autre que le francais. Le systeme doit detecter et rejeter ces requetes pour eviter un traitement incorrect.

*Exemples caracteristiques :*

* "Is there a train to Manchester?" (Anglais)
* "Quiero un billete para Madrid" (Espagnol)
* "Gibt es einen Zug nach Berlin?" (Allemand)
* "Voglio andare a Roma" (Italien)

**Classe UNKNOWN (Incomprehensible)**

Definition : Textes incomprehensibles, hors contexte, ou contenant uniquement du bruit textuel (sequences de caracteres sans sens, fautes de frappe majeures).

*Exemples caracteristiques :*

* "zzz"
* "lol lol lol"
* "asdfghjkl"
* "Merci pour le document" (hors contexte voyage)

## 2.3. Defis et Contraintes

La tache présente plusieurs defis techniques majeurs qui necessitent des solutions sophistiquées :

**1. Ambiguite semantique**

La frontiere entre TRIP et NOT\_TRIP peut etre subtile. Par exemple, "De Paris a Lyon" est un TRIP, mais "Y a-t-il des trains de Paris a Lyon ?" est un NOT\_TRIP. Le modele doit comprendre la nuance entre une demande de trajet et une demande d'information.

**2. Dataset**

Nous avons une tache de multi classification, il nous faut donc un datasets déjà labélisé ce qui pause un problème de trouvailles, les ia génératives comme chat gpt, claude ...etc pouvant créer des datasets incorrects, non varié.

**3. Variabilite linguistique**

Les utilisateurs peuvent formuler la meme intention de multiples facons : "De X a Y", "X vers Y", "Je veux aller de X a Y", "X - Y", "X Y". Le modele doit generaliser au-dela des formulations vues en entrainement.

**4. Detection multilingue**

Le systeme doit detecter avec precision les requetes en anglais, espagnol, allemand, italien, portugais, neerlandais, ...etc tout en etant robuste aux faux positifs (mots francais ressemblant a des mots etrangers).

**5. Extraction d'entites contextuelles**

Pour les requetes TRIP, le systeme doit extraire les villes de depart et destination, meme lorsqu'elles ne figurent pas dans un gazetteer predefined. Cela necessite une comprehension contextuelle plutot qu'une simple recherche de mots-cles.

# 3. Architecture des Transformers

## 3.1. Principe Fondamental

Les Transformers, introduits par Vaswani et al. (2017) dans l'article fondateur "Attention is All You Need", representent une revolution architecturale en NLP. Contrairement aux reseaux recurrents (RNN, LSTM) qui traitent le texte sequentiellement, les Transformers utilisent un mecanisme d'attention pour traiter tous les mots d'une phrase simultanement.

**Avantages cles des Transformers :**

* Parallelisation : Traitement simultane de tous les tokens
* Attention globale : Chaque mot peut "voir" tous les autres mots
* Pas de probleme de gradients evanescents comme dans les RNN
* Scalabilite : Performances s'ameliorent avec la taille du modele

## 3.2. Mecanisme d'Attention

Le mecanisme d'attention est le coeur des Transformers. Il permet au modele de determiner quels mots sont importants pour comprendre chaque mot de la phrase.

**Principe mathematique :**

L'attention calcule trois vecteurs pour chaque mot : Query (Q), Key (K), et Value (V). Le score d'attention entre deux mots est calcule par le produit scalaire de leurs vecteurs Q et K, normalise par softmax :

Attention(Q, K, V) = softmax(Q \* K^T / sqrt(d\_k)) \* V  
  
ou d\_k est la dimension des vecteurs Key

Cette formule permet au modele de "ponderer" l'importance de chaque mot pour comprendre le contexte d'un autre mot. Par exemple, dans "De Paris a Lyon", le modele apprendra que "Paris" est fortement lie a "De" (depart) et "Lyon" a "a" (destination).

**Multi-Head Attention :**

Les Transformers utilisent plusieurs "tetes" d'attention en parallele, permettant au modele de capturer differents types de relations : syntaxiques, semantiques, positionnelles. Chaque tete apprend a se concentrer sur un aspect different du texte.

## 3.3. CamemBERT pour le Francais

CamemBERT (Martin et al., 2019) est une adaptation de BERT (Bidirectional Encoder Representations from Transformers) specifiquement pre-entrainee sur un large corpus de textes francais.

**Caracteristiques de CamemBERT :**

* Architecture : 12 couches, 768 dimensions, 12 tetes d'attention
* Pre-entrainement : 138 GB de texte francais (OSCAR corpus)
* Tokenization : SentencePiece avec 32k tokens
* Objectif : Masked Language Modeling (MLM)

Le pre-entrainement MLM consiste a masquer aleatoirement 15% des mots d'une phrase et a entrainer le modele a les predire. Cela force le modele a developper une comprehension profonde de la syntaxe et semantique francaises.

**Avantages pour notre tache :**

* Comprehension contextuelle : Distingue "de Paris" (depart) vs "vers Paris" (destination)
* Robustesse aux variations : Generalise au-dela des patterns vus
* Detection semantique : Comprend "Je vais a X" = intention de voyage
* Representation vectorielle : Encode le sens plutot que la forme

# 4. Methodologie d'Implementation

## 4.1. Fine-Tuning pour Classification d'Intentions

Le fine-tuning consiste a adapter CamemBERT pre-entraine a notre tache specifique de classification en quatre classes. Cette approche tire parti des connaissances linguistiques deja acquises par le modele.

**Architecture du classifieur :**

On ajoute une couche de classification au-dessus de CamemBERT :

CamemBERT (pre-entraine)  
 |  
 v  
Representation [CLS] (768 dimensions)  
 |  
 v  
Dropout (0.4)  
 |  
 v  
Couche Dense (768 -> 4)  
 |  
 v  
Softmax  
 |  
 v  
[TRIP, NOT\_TRIP, NOT\_FRENCH, UNKNOWN]

**Hyperparametres optimises :**

* Learning rate : 8e-6 (tres faible pour fine-tuning)
* Batch size : 16 (equilibre GPU/convergence)
* Epochs : 6 (evite sur-apprentissage)
* Dropout : 0.4 (regularisation forte)
* Weight decay : 0.03 (regularisation L2)
* Warmup steps : 300 (montee progressive du LR)

**Gestion du desequilibre de classes :**

Pour compenser le desequilibre (TRIP 40%, NOT\_TRIP 30%, NOT\_FRENCH 20%, UNKNOWN 10%), on utilise une loss pondee :

class WeightedTrainer(Trainer):  
 def compute\_loss(self, model, inputs, return\_outputs=False):  
 labels = inputs.get("labels")  
 outputs = model(\*\*inputs)  
 logits = outputs.get("logits")  
   
 # Poids calcules par compute\_class\_weight  
 weight\_tensor = torch.tensor(class\_weights).to(logits.device)  
 loss\_fct = nn.CrossEntropyLoss(weight=weight\_tensor)  
 loss = loss\_fct(logits, labels)  
   
 return (loss, outputs) if return\_outputs else loss

Les poids sont calcules automatiquement par sklearn.compute\_class\_weight pour donner plus d'importance aux classes minoritaires.

## 4.2. Named Entity Recognition (NER)

Pour les requetes classifiees TRIP, le systeme doit extraire les villes de depart et destination. Nous utilisons une approche Token Classification avec CamemBERT.

**Format BIO :**

Les annotations sont converties en format BIO (Begin, Inside, Outside) :

* O : Token n'appartient a aucune entite
* B-Departure : Debut d'une ville de depart
* I-Departure : Continuation d'une ville de depart
* B-Destination : Debut d'une ville de destination
* I-Destination : Continuation d'une ville de destination

*Exemple d'annotation :*

Texte : "Je vais de Saint-Denis a Paris"  
  
Tokens: Je vais de Saint - Denis a Paris  
Labels: O O O B-Dep I-Dep I-Dep O B-Dest

Cette approche permet de gerer les villes composees (Saint-Denis, Les Sables-d'Olonne) et les variations orthographiques.

**Architecture NER :**

CamemBERT (pre-entraine)  
 |  
 v  
Representations par token (768 dimensions)  
 |  
 v  
Couche Dense (768 -> 5)  
 |  
 v  
Softmax par token  
 |  
 v  
[O, B-Departure, I-Departure, B-Destination, I-Destination]

## 4.3. Post-Processing Heuristique

Bien que CamemBERT soit performant, certaines erreurs systematiques necessitent des regles de correction heuristiques :

**Regle 1 : Detection de langue**

Si le texte contient des marqueurs linguistiques etrangers (ex: "I ", "the ", "is "), forcer la classe NOT\_FRENCH meme si le modele a predit TRIP.

**Regle 2 : Mots-cles NOT\_TRIP**

Si le texte contient "merci", "email", "document", "rapport" sans contexte de voyage, corriger vers NOT\_TRIP ou UNKNOWN.

**Regle 3 : Format "Ville1 Ville2"**

Si le texte est compose uniquement de deux noms propres (majuscules), c'est probablement un TRIP (ex: "Paris Lyon").

Ces regles ameliorent la precision de 2-3% en corrigeant les cas limites mal geres par le modele.

# 5. Implementation Google Colab

## 5.1. Architecture Complete du Systeme

L'implementation sous Google Colab permet d'utiliser des GPUs gratuits pour accelerer l'entrainement des modeles Transformers. Voici l'architecture complete :

**Pipeline d'entrainement :**

1. Configuration GPU  
 - Verification disponibilite CUDA  
 - Installation packages (transformers, datasets, evaluate)  
  
2. Chargement Donnees  
 - Montage Google Drive  
 - Lecture train\_set.csv (8000 exemples)  
 - Lecture test\_set.csv (2000 exemples)  
 - Parsing annotations JSON  
  
3. Preprocessing  
 - Nettoyage textes  
 - Encodage labels (LabelEncoder)  
 - Conversion format HuggingFace Dataset  
 - Tokenization avec CamemBERT tokenizer  
  
4. Fine-Tuning Intent Classification  
 - Chargement CamemBERT pre-entraine  
 - Calcul class weights  
 - Entrainement avec WeightedTrainer  
 - Evaluation (accuracy, F1 macro, F1 per class)  
 - Sauvegarde modele  
  
5. Fine-Tuning NER  
 - Conversion annotations en format BIO  
 - Chargement CamemBERT pour Token Classification  
 - Entrainement avec seqeval metrics  
 - Evaluation par entite (Departure, Destination)  
 - Sauvegarde modele  
  
6. Inference Pipeline  
 - Chargement modeles entraines  
 - Creation pipelines HuggingFace  
 - Tests sur exemples

## 5.2. Code Google Colab Complet

# Conclusion

Cette revue de litterature a presente une analyse complete du projet de classification d'intentions multi-categorie base sur l'architecture Transformers, specifiquement CamemBERT pour le francais.