

# Travel Order Resolver

## Rapport Scientifique

Ludovic Tagnon

2026-02-12

### Table des matières

<b>Page de garde</b>	<b>4</b>
<b>Instructions de rendu PDF</b>	<b>4</b>
<b>Travel Order Resolver - Rapport scientifique (version longue)</b>	<b>4</b>
Resume	4
1. Introduction	5
1.1 Contexte	5
1.2 Objectif scientifique	5
2. Problematique et contraintes	5
2.1 Formulation du probleme	5
2.2 Contraintes systeme	5
3. Questions de recherche	6
4. Strategie de realisation	6
4.1 Principe directeur	6
4.2 Justification strategique	6
5. Jeux de donnees	6
5.1 Donnees synthetiques	6
5.2 Donnees manuelles	7
5.3 Donnees ferroviaires	7
6. Architecture technique	7
6.1 Vue d'ensemble	7
6.2 Backend NLP multi-mode	7
6.3 Industrialisation	8
7. Methodologie experimentale	8
7.1 Metriques	8
7.2 Protocole	8
7.3 Reproductibilite	8
8. Approche rule-based (baseline principale)	8
8.1 Conception	8
8.2 Pourquoi cette approche marche ici	9
8.3 Resultats	9
9. Baseline ML classique (char n-grams + LinearSVC)	9
9.1 Objectif de ce baseline	9
9.2 Resultats	9
9.3 Lecture critique	10
10. Benchmarks spaCy et CamemBERT non fine-tune	10
10.1 spaCy	10
10.2 CamemBERT embeddings figes + SVC	10

10.3 Analyse . . . . .	10
11. CamemBERT fine-tune . . . . .	10
11.1 v1 rapide (preuve de concept) . . . . .	10
11.2 v2 renforcee . . . . .	11
11.3 Interpretation . . . . .	11
12. Pathfinding . . . . .	11
12.1 Modele de graphe . . . . .	11
12.2 Resolution des noms . . . . .	12
12.3 Resultats . . . . .	12
13. Evaluation end-to-end . . . . .	12
13.1 Lot manuel 120 (rule-based) . . . . .	12
13.2 Lot manuel 120 (CamemBERT v2) . . . . .	12
13.3 Gold manuel 120 . . . . .	12
14. Analyse d'erreurs . . . . .	13
14.1 Typologie des erreurs observees . . . . .	13
14.2 Erreurs utiles (ratés) . . . . .	13
14.3 Ce qui a marche . . . . .	13
15. Challenges techniques et strategie . . . . .	13
15.1 Challenge 1: robustesse linguistique . . . . .	13
15.2 Challenge 2: preuve de qualite . . . . .	13
15.3 Challenge 3: maitrise du risque projet . . . . .	13
16. Discussion scientifique . . . . .	14
16.1 Sur la performance . . . . .	14
16.2 Sur la transferabilite . . . . .	14
16.3 Sur le cout . . . . .	14
17. Validite et limites . . . . .	14
17.1 Menaces de validite interne . . . . .	14
17.2 Menaces de validite externe . . . . .	14
17.3 Limites techniques actuelles . . . . .	14
18. Reproductibilite et artefacts . . . . .	15
18.1 Scripts clés . . . . .	15
18.2 Artefacts resultats . . . . .	15
18.3 Bundle . . . . .	15
19. Etat actuel et plan avant soutenance . . . . .	15
19.1 Etat des points critiques . . . . .	15
19.2 Travail restant recommandé . . . . .	15
20. Conclusion . . . . .	15
Annexe A - Commandes de reproduction . . . . .	16
A.1 Verification globale . . . . .	16
A.2 Baselines NLP . . . . .	16
A.3 CamemBERT . . . . .	16
A.4 Evaluation manuelle et dashboard . . . . .	16
Annexe B - Tableau comparatif principal . . . . .	16
Annexe C - Journal de decisions (synthèse) . . . . .	16
C.1 Decision 1 . . . . .	16
C.2 Decision 2 . . . . .	17
C.3 Decision 3 . . . . .	17
C.4 Decision 4 . . . . .	17
C.5 Decision 5 . . . . .	17
Annexe D - Recommandations de mise en page PDF (objectif ~20 pages) . . . . .	17
21. Etudes de cas détaillées . . . . .	17
21.1 Cas A - Phrase standard . . . . .	17
21.2 Cas B - Destination avant origine . . . . .	18
21.3 Cas C - Ville intermédiaire explicite . . . . .	18

21.4 Cas D - INVALID pur . . . . .	18
21.5 Cas E - Bruit orthographique . . . . .	19
21.6 Cas F - Ambiguite prenom/ville . . . . .	19
21.7 Cas G - Exemple pipeline complet . . . . .	19
21.8 Cas H - Meme score E2E, causes differente . . . . .	19
22. Comparaison quantitative multi-niveaux . . . . .	20
22.1 Niveau 1 - NLP sur test synthetique . . . . .	20
22.2 Niveau 2 - NLP sur gold manuel . . . . .	20
22.3 Niveau 3 - End-to-end . . . . .	20
23. Retours d'experience: erreurs, ratés, pivots . . . . .	21
23.1 Raté 1 - Surconfiance dans un baseline neurale "plug-and-play" . . . . .	21
23.2 Raté 2 - Croire qu'un score unique suffit . . . . .	21
23.3 Raté 3 - Relecture manuelle non priorisee . . . . .	21
23.4 Reussite 1 - Approche incrementaliste . . . . .	21
23.5 Reussite 2 - Separation stricte NLP / pathfinding . . . . .	22
23.6 Reussite 3 - Dashboard comparatif . . . . .	22
24. Couts, risques et compromis . . . . .	22
24.1 Compromis performance vs maintenance . . . . .	22
24.2 Compromis court terme vs long terme . . . . .	22
24.3 Registre de risques . . . . .	22
25. Protocole de soutenance technique . . . . .	23
25.1 Narratif conseille . . . . .	23
25.2 Questions jury anticipees . . . . .	23
26. Extension scientifique envisagee . . . . .	23
26.1 Experimentes non realises mais prioritaires . . . . .	23
26.2 Proposition de protocole post-projet . . . . .	23
26.3 Critere de "production readiness" . . . . .	23
27. Conclusion et positionnement final . . . . .	23
Annexe E - Catalogue d'erreurs observees (synthese) . . . . .	24
E.1 Erreurs NLP historiquement frequentes (avant stabilisation) . . . . .	24
E.2 Erreurs residuelles apres stabilisation . . . . .	24
E.3 Actions correctives associees . . . . .	24
Annexe F - Chronologie compacte du projet . . . . .	24
28. Details algorithmiques . . . . .	24
28.1 Normalisation textuelle . . . . .	24
28.2 Indexation des lieux . . . . .	25
28.3 Matching fuzzy . . . . .	25
28.4 Extraction guidee par cues . . . . .	26
28.5 Pathfinding BFS . . . . .	26
28.6 Mapping ville -> stop area . . . . .	26
29. Analyse de complexite et passage a l'echelle . . . . .	27
29.1 Cote NLP rule-based . . . . .	27
29.2 Cote Camembert v2 . . . . .	27
29.3 Cote pathfinding . . . . .	27
30. Analyse de robustesse . . . . .	27
30.1 Robustesse au bruit lexical . . . . .	27
30.2 Robustesse aux structures non canoniques . . . . .	28
30.3 Robustesse INVALID . . . . .	28
30.4 Robustesse end-to-end . . . . .	28
31. Discussion sur la qualite des preuves . . . . .	28
31.1 Ce qui est solide . . . . .	28
31.2 Ce qui reste fragile . . . . .	28
31.3 Niveau de confiance . . . . .	28
32. Gouvernance technique et qualite logicielle . . . . .	29

32.1 Hygiene de code . . . . .	29
32.2 Traçabilite . . . . .	29
32.3 Non regressions . . . . .	29
33. Feuille de route vers version finale PDF . . . . .	29
33.1 Structure cible 20 pages . . . . .	29
33.2 Elements visuels a inserer . . . . .	29
33.3 Texte deja pret . . . . .	30

## Page de garde

**Projet:** Travel Order Resolver

**Formation:** EPITECH

**Auteur principal:** Ludovic Tagnon

**Version:** Finale longue

**Date:** 2026-02-12

\newpage

## Instructions de rendu PDF

- Format cible: ~20 pages.
- Interligne recommande: 1.15.
- Marges recommandees: 2.5 cm.
- Numeroter les tableaux et les figures.
- Inserer les figures annonces dans `docs/figures/`.

\newpage

## Travel Order Resolver - Rapport scientifique (version longue)

### Resume

Ce document decrit de maniere scientifique la conception, l'implementation et l'evaluation d'un systeme de resolution d'ordres de voyage ferroviaire en langage naturel. Le projet vise a transformer une phrase libre de type chatbot en deux sorties operationnelles: (1) un triplet NLP `id,origin,destination` ou `id,INVALID`, et (2) un itineraire exploitable par un module de pathfinding.

Le travail a ete mene en plusieurs iterations, avec une logique de reduction de risque: baseline rule-based robuste, baseline ML de reference, essais spaCy et CamemBERT, puis fine-tuning CamemBERT complet. Le systeme final supporte deux backends NLP (`rule-based` et `camembert-ft`) et un module pathfinding valide sur des donnees GTFS.

Sur le split test synthetique, la baseline rule-based atteint 99.3% d'accuracy, CamemBERT fine-tune v2 atteint 97.3%, spaCy 69.3%, et le baseline ML 41.8%. Sur le gold set manuel (120 phrases), le rule-based atteint 120/120 et CamemBERT v2 119/120. Le pathfinding atteint 30/30 sur l'echantillon de validation dedie.

Au-delà des resultats, ce rapport documente la demarche strategique, les erreurs, les impasses, les compromis techniques et les limites de validite du protocole. L'objectif est de produire une trace defendable en soutenance, reproductible et exploitable pour une poursuite de projet.

# 1. Introduction

## 1.1 Contexte

Le sujet "Travel Order Resolver" demande de traiter des ordres de voyage en langage naturel pour en extraire un depart et une destination, puis de calculer un trajet ferroviaire coherent a partir de donnees SNCF/GTFS. Le projet est volontairement transverse: NLP, qualite de donnees, algorithmique de graphe, reproductibilite experimentale, et industrialisation.

Dans un cadre academique, le risque principal est de concentrer l'effort sur un modele "avance" sans stabiliser les fondamentaux (I/O, robustesse, protocole d'evaluation, outillage reproductible). La strategie adoptee ici est l'inverse: stabiliser rapidement un pipeline mesurable puis complexifier par paliers.

## 1.2 Objectif scientifique

L'objectif n'est pas uniquement de "faire marcher" un parseur de phrases, mais de repondre a une question technique:

"Quel compromis entre robustesse, cout de dev, cout de calcul et qualite predicitive permet une livraison defendable dans les contraintes du projet ?"

Cette question est traitee par comparaison experimentale de plusieurs familles d'approches:

- rule-based deterministe,
- ML classique (char n-grams + LinearSVC),
- spaCy (NER/cues),
- CamemBERT embeddings figes + SVC,
- CamemBERT fine-tune end-to-end.

# 2. Problematique et contraintes

## 2.1 Formulation du probleme

Entree: une ligne `id,phrase`.

Sortie NLP:

- cas valide: `id,origin,destination`
- cas invalide/hors sujet: `id,INVALID`,

Sortie pathfinding:

- sequence de gares `id,step0,step1,...,stepN`

La difficulte majeure reside dans l'extraction robuste de l'origine et de la destination dans des phrases bruitées:

- fautes de frappe,
- absence d'accents,
- prepositions ambiguës,
- presence de villes intermediaires,
- mots qui peuvent etre prenoms ou lieux.

## 2.2 Contraintes systeme

- encodage UTF-8,
- execution CLI (pas de dependance web obligatoire),

- reproductibilite locale,
- evaluation explicite sur splits train/dev/test et lot manuel,
- rendu operationnel testable par des scripts.

### 3. Questions de recherche

Nous avons organise le travail autour de 5 questions:

1. Une approche rule-based bien ingenieriee suffit-elle pour un niveau de performance eleve ?
2. Un baseline ML simple peut-il depasser le rule-based dans ce contexte de donnees ?
3. Quelle valeur incrementale apportent spaCy et CamemBERT sans fine-tuning ?
4. Le fine-tuning CamemBERT apporte-t-il un gain significatif et defendable ?
5. Le gain NLP se traduit-il vraiment en gain end-to-end une fois le pathfinding applique ?

### 4. Strategie de realisation

#### 4.1 Principe directeur

Le plan a suivi une logique de "sprints de preuve":

- Sprint A: pipeline minimal complet (NLP + pathfinding + metriques)
- Sprint B: stabilisation des donnees et validation manuelle
- Sprint C: baselines ML/NLP alternatives pour comparaison
- Sprint D: fine-tuning CamemBERT et integration multi-backend
- Sprint E: industrialisation de rendu (snapshot, bundle, dashboard)

#### 4.2 Justification strategique

Cette approche limite le risque de blocage:

- si le modele avance echoue, la livraison reste fonctionnelle,
- chaque palier produit des evidences (fichiers + scripts + metriques),
- la soutenance peut montrer un cheminement scientifique et non un resultat "boite noire".

### 5. Jeux de donnees

#### 5.1 Donnees synthetiques

Le coeur des entrainements/evaluations repose sur:

- datasets/train\_input.txt / datasets/train\_output.txt
- datasets/dev\_input.txt / datasets/dev\_output.txt
- datasets/test\_input.txt / datasets/test\_output.txt

Statistiques principales:

- train: 8000 lignes (6763 valides, 1237 invalides),
- dev: 1000 lignes (867 valides, 133 invalides),
- test: 1000 lignes (855 valides, 145 invalides).

Longueur moyenne de phrase:

- train: 6.84 tokens,
- dev: 6.92 tokens,

- test: 7.07 tokens.

## 5.2 Donnees manuelles

Pour sortir du cadre purement synthetique:

- `datasets/manual/input_starter.csv` (120 phrases),
- `datasets/manual/output_gold_120.csv` (reference finale),
- `datasets/manual/corrections_120.csv` (22 lignes prioritaires).

Validation de coherence:

- `input_lines=120, output_lines=120,`
- `valid=115, invalid=5,`
- `pending=0, malformed=0.`

## 5.3 Donnees ferroviaires

- `stops.xlsx` (liste gares/identifiants),
- GTFS (`stops.txt, stop_times.txt`) pour construire le graphe,
- index gares: `data/stops_index.json`.

Graphe derive (`data/graph.json`):

- `node_count=3547,`
- `edge_count=11430.`

# 6. Architecture technique

## 6.1 Vue d'ensemble

Le systeme suit un pipeline lineaire:

1. lecture des phrases,
2. extraction NLP,
3. resolution des noms vers IDs de gares,
4. pathfinding,
5. generation des sorties et metriques.

Composants principaux:

- `src/travel_order_resolver.py` (NLP rule-based),
- `scripts/pathfind.py` (resolution et plus court chemin),
- `scripts/run_pipeline.py` (orchestration complete),
- `scripts/evaluate*.py` (metriques),
- `scripts/run_snapshot.py` (consolidation).

## 6.2 Backend NLP multi-mode

Le pipeline supporte:

- `rule-based` (defaut),
- `camembert-ft` (fine-tune v2).

Cela permet un benchmark strict a pathfinding constant.

### 6.3 Industrialisation

- tests unitaires sous `tests/`,
- CI GitHub Actions,
- cibles Makefile (`train/eval/snapshot/bundle`),
- bundle livrable hashé (`deliverables/submitmission_bundle/manifest.json`).

## 7. Methodologie experimentale

### 7.1 Metriques

Metriques principales:

- accuracy globale,
- invalid\_accuracy (classification hors trajet),
- valid\_precision, valid\_recall, valid\_f1,
- origin\_accuracy, destination\_accuracy.

Pour l'end-to-end:

- taux NLP valide,
- succes pathfinding conditionnel,
- succes global pipeline.

### 7.2 Protocole

- entrainement sur train,
- selection/diagnostic sur dev,
- resultat final sur test,
- verification de generalisation sur lot manuel gold.

### 7.3 Reproductibilite

Commandes centrales:

- `make test`
- `make train-ml`
- `make spacy-camembert-bench`
- `make train-camembert-ft-v2`
- `make camembert-ft-v2-bench`
- `make manual-gold-eval-camembert-v2`
- `make snapshot`
- `make bundle`

## 8. Approche rule-based (baseline principale)

### 8.1 Conception

Le module rule-based combine:

- normalisation forte (casse, accents, ponctuation),
- reconnaissance de variantes de lieux,
- heuristiques de cues linguistiques (de/depuis/vers/pour...),

- fuzzy matching (distance d'édition avec seuil adaptatif),
- fallback sur extraction de lieux sans cues explicites.

## 8.2 Pourquoi cette approche marche ici

Le domaine est relativement contraint:

- vocabulaire de villes connu,
- structure de phrase semi-guidée,
- forte redondance des patterns.

Une logique déterministe bien calibrée capture efficacement ce régime.

## 8.3 Résultats

Source: `reports/metrics.json`.

Split	Accuracy	Valid F1	Invalid Accuracy
Train	0.983	0.987	1.000
Dev	0.991	0.993	1.000
Test	0.993	0.995	1.000

Interpretation:

- excellente généralisation train->dev->test,
- stabilité des cas INVALID,
- faible variance entre splits.

## 9. Baseline ML classique (char n-grams + LinearSVC)

### 9.1 Objectif de ce baseline

Le baseline ML n'a pas été introduit pour remplacer le rule-based immédiatement, mais pour fournir:

- un point de comparaison quantitatif,
- une base d'analyse d'erreurs,
- un jalon scientifique pour justifier les évolutions.

### 9.2 Résultats

Source: `reports/ml_metrics.json`.

Split	Accuracy	Valid F1	Invalid Accuracy
Train	0.703	0.656	0.950
Dev	0.404	0.328	0.887
Test	0.418	0.338	0.876

### 9.3 Lecture critique

Ce modèle sur-apprend partiellement le train et généralise mal. Il confond fréquemment:

- origine/destination inversées,
- villes intermédiaires prises pour destination,
- faux positifs sur phrases ambiguës.

Exemples d'erreurs:

- "trains reims angers" -> inversion Reims/Angers,
- phrase avec "en passant par Marseille" -> destination prédites sur l'intermédiaire.

Conclusion: baseline utile pour apprendre, insuffisante pour livraison.

## 10. Benchmarks spaCy et CamemBERT non fine-tune

### 10.1 spaCy

Source: `reports/spacy_camembert_metrics.json`.

Split	Accuracy	Valid F1
Dev	0.694	0.775
Test	0.693	0.771

### 10.2 CamemBERT embeddings figes + SVC

Source: `reports/spacy_camembert_metrics.json`.

Split	Accuracy	Valid F1
Dev	0.482	0.409
Test	0.498	0.418

### 10.3 Analyse

- spaCy dépasse clairement la baseline ML,
- CamemBERT fige ne suffit pas dans cette configuration,
- ni spaCy ni CamemBERT fige n'atteignent le rule-based.

Ce palier a servi de "raté utile": il a confirmé que la valeur venait du fine-tuning, pas du simple usage d'embeddings pré-entraînés.

## 11. CamemBERT fine-tune

### 11.1 v1 rapide (preuve de concept)

Configuration v1:

- 4000 lignes,
- 1 epoch,
- batch 16,
- max\_length 64,

- deux classifieurs séparés (origin/destination).

Réultats v1 (`reports/camembert_finetune_metrics.json`):

- dev accuracy 0.733, valid\_f1 0.753,
- test accuracy 0.735, valid\_f1 0.751.

Apport: confirmation que le fine-tuning fonctionne, mais encore loin du rule-based.

## 11.2 v2 renforcee

Configuration v2:

- 8000 lignes train,
- 2 epochs,
- lr 2e-5,
- batch 16,
- max\_length 64,
- seed 42.

Metadata entraînement:

- origin best dev accuracy 0.991,
- destination best dev accuracy 0.989.

Réultats v2 (`reports/camembert_finetune_v2_metrics.json`):

Split	Accuracy	Valid F1	Origin Acc	Destination Acc
Dev	0.981	0.978	0.990	0.987
Test	0.973	0.968	0.986	0.982

## 11.3 Interpretation

Le saut v1 -> v2 est majeur:

- +23.8 points d'accuracy test,
- +21.7 points de valid\_f1 test.

Ce résultat montre que l'approche neurale est performante une fois:

- le volume de train augmente,
- les hyperparamètres sont stabilisés,
- le mode end-to-end est assumé.

## 12. Pathfinding

### 12.1 Modèle de graphe

Le réseau est représenté par un graphe orienté des `StopArea`. La recherche de chemin est traitée par un algorithme de plus court chemin de type BFS (non pondéré).

## 12.2 Resolution des noms

La resolution texte -> stop\_ids combine:

- exact match,
- prefix match,
- fuzzy match,
- gestion de variantes (Saint/St) et bruit encodage.

## 12.3 Resultats

reports/pathfinding\_metrics.txt:

- total=30, correct=30, accuracy=1.000.

Sur ce perimetre, le pathfinding n'est pas le facteur limitant. La precision globale depend surtout du module NLP.

## 13. Evaluation end-to-end

### 13.1 Lot manuel 120 (rule-based)

reports/e2e\_manual\_120\_summary.json:

- NLP valide: 115/120,
- path valide sur NLP valide: 115/115,
- succes global: 115/120 (95.8%).

### 13.2 Lot manuel 120 (CamemBERT v2)

reports/e2e\_manual\_120\_camembert\_v2\_summary.json:

- NLP valide: 115/120,
- path valide sur NLP valide: 115/115,
- succes global: 115/120 (95.8%).

Observation: les deux backends atteignent le meme score E2E sur ce lot. Cela ne signifie pas qu'ils font exactement les memes erreurs, mais que leur impact final est equivalent a ce niveau de granularite.

### 13.3 Gold manuel 120

Dashboard consolide: reports/manual\_gold\_dashboard.json.

NLP:

- rule-based: 120/120 (1.000),
- CamemBERT v2: 119/120 (0.992),
- ML baseline: 67/120 (0.558).

E2E:

- rule-based: 115/120 (95.8%),
- CamemBERT v2: 115/120 (95.8%).

## 14. Analyse d'erreurs

### 14.1 Typologie des erreurs observees

1. Inversion origine/destination sur structures nominales tres courtes.
2. Confusion destination/intermediaire avec pattern "en passant par".
3. Ambiguites lexicales (pronoms, noms communs proches de villes).
4. Cas INVALID trompeurs (phrases proches du domaine sans intention trajet).

### 14.2 Erreurs utiles (ratés)

Quelques essais n'ont pas donne le resultat espere, mais ont clarifie la direction:

- CamemBERT embeddings figes + SVC: performance mediocre, confirme que la phase task-specific est critique.
- Baseline ML classique: bon outil pedagogique, mais generalisation insuffisante.
- Premiers essais de revue manuelle: priorisation indispensable, sinon effort trop diffuse.

### 14.3 Ce qui a marche

- priorisation des cas actionnables (22/120),
- separation claire NLP vs pathfinding,
- benchmarks outilles et repetables,
- comparative dashboard pour arbitrage objectif.

## 15. Challenges techniques et strategie

### 15.1 Challenge 1: robustesse linguistique

Probleme: bruit orthographique, accents absents, ordre syntaxique variable.

Reponse:

- normalisation aggressive,
- fuzzy matching avec seuil adapte,
- cues linguistiques + fallbacks.

### 15.2 Challenge 2: preuve de qualite

Probleme: un score isole n'explique pas la fiabilite reelle.

Reponse:

- multiplicité de métriques,
- évaluation multi-splits,
- dashboard comparatif,
- lot manuel gold.

### 15.3 Challenge 3: maîtrise du risque projet

Probleme: une approche purement "modèle avancé" était risquée en délai.

Reponse stratégique:

- baseline rule-based d'abord,

- complexification progressive,
- conservation d'un pipeline livrable a chaque etape.

## 16. Discussion scientifique

### 16.1 Sur la performance

Le resultat le plus fort est double:

- un rule-based extremement solide dans ce domaine,
- un CamemBERT fine-tune v2 capable d'approcher ce niveau.

Ce constat est interessant pedagogiquement: dans un domaine semi-constraint, une ingenierie de regles peut rivaliser avec des modeles lourds, au moins a court terme.

### 16.2 Sur la transferabilite

Le risque principal est la transferabilite hors distribution synthetique:

- changement de style utilisateur,
- villes hors dictionnaire,
- phrases plus longues et multi-intentions.

D'où l'importance d'augmenter le jeu manuel et de maintenir une boucle d'erreur active.

### 16.3 Sur le cout

Comparaison qualitative:

- rule-based: cout GPU nul, debuggable, maintenance manuelle.
- CamemBERT v2: meilleur potentiel de generalisation, cout entrainement/inference plus eleve.

## 17. Validite et limites

### 17.1 Menaces de validite interne

- datasets synthetiques potentiellement proches du generateur,
- taille du lot manuel encore modeste (120),
- simple annotation majoritaire (pas de double aveugle complet).

### 17.2 Menaces de validite externe

- couverture linguistique francaise reelle non exhaustive,
- test utilisateur en conditions reelles non realise.

### 17.3 Limites techniques actuelles

- pathfinding non pondere temporellement (pas de cout horaire/waiting),
- pas de gestion avancee des multi-etapes demandees explicitement,
- pas de calibration probabiliste explicite en sortie NLP.

## 18. Reproductibilite et artefacts

### 18.1 Scripts clés

- extraction/bench: scripts/evaluate.py, scripts/evaluate\_ml.py, scripts/evaluate\_camembert\_finetune.py
- E2E: scripts/evaluate\_end\_to\_end.py, scripts/run\_pipeline.py
- consolidation: scripts/run\_snapshot.py, scripts/run\_manual\_gold\_eval.py
- rendu: scripts/build\_submission\_bundle.py

### 18.2 Artefacts résultats

- reports/metrics.json
- reports/ml\_metrics.json
- reports/spacy\_camembert\_metrics.json
- reports/camembert\_finetune\_metrics.json
- reports/camembert\_finetune\_v2\_metrics.json
- reports/manual\_gold\_dashboard.json
- reports/snapshot.json, reports/snapshot.md

### 18.3 Bundle

Le bundle deliverables/submission\_bundle/ contient les fichiers cibles et un manifeste SHA256. C'est l'unité de rendu reproductible.

## 19. Etat actuel et plan avant soutenance

### 19.1 Etat des points critiques

- pipeline complet: fait,
- comparaison NLP multi-approches: faite,
- dashboard comparatif gold: fait,
- validations unitaires/CI: faites,
- rapport long scientifique: en cours de finalisation editoriale.

### 19.2 Travail restant recommandé

1. augmenter le lot manuel annoté (objectif > 300 phrases),
2. vérifier les écarts rule-based vs CamemBERT sur des cas réels hors distribution,
3. éventuellement ajouter un ranking confiance pour la sortie NLP,
4. produire la version PDF finale avec schéma d'architecture et tableaux numérotés.

## 20. Conclusion

Le projet atteint un niveau de maturité technique défendable:

- solution opérationnelle complète NLP + pathfinding,
- niveau de performance élevé et mesuré,
- documentation reproductible,
- preuves quantitatives et qualitatives sur plusieurs familles de modèles.

La contribution principale est moins "un modèle" qu'une architecture et une démarche:

- commencer par stabiliser,

- mesurer systematiquement,
- complexifier seulement quand les preuves l'exigent,
- conserver des traces experimentales a chaque pivot.

Le resultat est un systeme qui peut etre soutenu techniqueument, audite, reproduit, et prolonge.

---

## Annexe A - Commandes de reproduction

### A.1 Verification globale

```
make test
make snapshot
make bundle
```

### A.2 Baselines NLP

```
make benchmarks
make ml-benchmarks
make spacy-camembert-bench
```

### A.3 CamemBERT

```
make train-camembert-ft
make camembert-ft-bench
make train-camembert-ft-v2
make camembert-ft-v2-bench
```

### A.4 Evaluation manuelle et dashboard

```
make manual-gold-eval
make manual-gold-eval-camembert-v2
make e2e-camembert-ft-v2
```

## Annexe B - Tableau comparatif principal

Modele	Test Accuracy	Test Valid F1	Gold 120 Accuracy
Rule-based	0.993	0.995	1.000
CamemBERT FT v2	0.973	0.968	0.992
CamemBERT FT v1	0.735	0.751	n/a
spaCy	0.693	0.771	n/a
ML baseline	0.418	0.338	0.558
CamemBERT fige + SVC	0.498	0.418	n/a

## Annexe C - Journal de decisions (synthese)

### C.1 Decision 1

Choix initial rule-based avant modele lourd. Raison: reduction du risque de non-livraison. Impact: pipeline complet disponible tres tot.

## C.2 Decision 2

Conserver un baseline ML faible mais instrumente. Raison: reference scientifique et analyse d'erreurs. Impact: justification claire des pivots ulterieurs.

## C.3 Decision 3

Passer de CamemBERT fige a fine-tune end-to-end. Raison: performance insuffisante du setup fige. Impact: gain majeur (test accuracy 0.498 -> 0.973).

## C.4 Decision 4

Integrer un backend multi-mode dans le pipeline. Raison: comparer a pathfinding constant et preparer la soutenance. Impact: evaluation E2E symetrique rule-based vs CamemBERT.

## C.5 Decision 5

Ajouter dashboard gold avec leaderboard. Raison: arbitrage de modele lisible et rapide. Impact: argumentaire de soutenance plus solide.

## Annexe D - Recommandations de mise en page PDF (objectif ~20 pages)

Pour atteindre un rendu academique de ~20 pages sans "blabla" inutile:

1. inserer un schema d'architecture pleine largeur (pipeline + modules),
2. inserer un schema de flux de donnees (input -> NLP -> triplet -> path),
3. ajouter 3 tableaux numerotes (metriques synthetiques, gold set, E2E),
4. ajouter 1 section "Etude de cas" avec 10 exemples commentes,
5. conserver interligne 1.15 et police standard academique.

Ces ajouts visuels convertissent naturellement cette version longue en un document final de taille attendue.

## 21. Etudes de cas detailles

Cette section vise a documenter la comprehension fine du comportement systeme. L'objectif n'est pas de multiplier les exemples artificiels, mais de montrer pourquoi une sortie est correcte, pourquoi une erreur se produit, et quels leviers techniques permettent de corriger.

### 21.1 Cas A - Phrase standard

Phrase: "je voudrais aller de Toulouse a Bordeaux"

Attendu:

- origin: Toulouse
- destination: Bordeaux

Analyse:

- les cues "de" et "a" delimitent clairement l'intention,
- aucun terme intermediaire,
- aucune ambiguïté lexicale.

Comportement:

- rule-based: correct,
- ML baseline: généralement correct sur ce pattern,
- Camembert v2: correct.

Enseignement: ce type de phrase ne discrimine pas les modèles, il valide surtout la chaîne I/O.

## 21.2 Cas B - Destination avant origine

Phrase: "comment me rendre à Port-Boulet depuis Tours ?"

Difficulté:

- destination exprimée avant l'origine,
- préposition "depuis" interprétable selon le contexte.

Comportement:

- rule-based: correct (grâce aux indices),
- baseline ML: variable,
- Camembert v2: correct.

Enseignement: la robustesse aux permutations syntaxiques est un critère critique.

## 21.3 Cas C - Ville intermédiaire explicite

Phrase: "départ Montpellier destination Port-Boulet en passant par Marseille"

Attendu:

- origin: Montpellier
- destination: Port-Boulet
- Marseille est une ville intermédiaire, pas la destination finale.

Erreur observée historiquement:

- baseline ML prédit parfois Marseille comme destination.

Raison:

- approximation statistique locale,
- difficultés à modéliser les dépendances longues sans représentation adaptée.

Enseignement: la gestion des intermédiaires est un excellent test de compréhension structurelle.

## 21.4 Cas D - INVALID pur

Phrase: "horaires pour demain"

Attendu: INVALID.

Comportement:

- rule-based: correct,
- Camembert v2: correct,
- baseline ML: tendance aux faux positifs sur certains contextes proches du domaine.

Enseignement: la détection INVALID doit être évaluée séparément (`invalid_accuracy`).

## **21.5 Cas E - Bruit orthographique**

Phrase type: "saint-nazajre"

Attendu: normalisation vers "Saint-Nazaire".

Levier technique:

- normalisation + distance d'édition avec seuil dynamique.

Enseignement: les heuristiques de nettoyage donnent un gain majeur à cout faible.

## **21.6 Cas F - Ambiguité nom/ville**

Phrase type: "aller à Tours voir mon ami Albert"

Risque:

- confusion entre entité personne et lieu.

Comportement attendu:

- extraire Tours comme destination si la structure indique un trajet,
- ignorer Albert comme destination selon le contexte.

Enseignement: sans annotation explicite de rôle sémantique, l'ordre et les prépositions restent essentiels.

## **21.7 Cas G - Exemple pipeline complet**

Entrée:

- `id=2, phrase sample.`

Sortie NLP Camembert v2:

- `Tours,Port-Boulet`

Sortie pathfinding:

- `Tours -> Saumur -> Port Boulet`

Interprétation:

- NLP correct,
- résolution gares correcte,
- chemin cohérent.

Enseignement: le gain NLP n'a de valeur que s'il se propage jusqu'à la sortie métier.

## **21.8 Cas H - Même score E2E, causes différentes**

Observation:

- rule-based et Camembert v2 ont 115/120 en E2E sur le lot manuel.

Hypothèse:

- les 5 échecs ne sont pas nécessairement les mêmes IDs.

Impact:

- important pour une fusion future (ensemble model): deux modeles de score égal peuvent être complémentaires.

## 22. Comparaison quantitative multi-niveaux

### 22.1 Niveau 1 - NLP sur test synthétique

Modele	Accuracy	Valid F1
Rule-based	0.993	0.995
Camembert FT v2	0.973	0.968
Camembert FT v1	0.735	0.751
spaCy	0.693	0.771
Camembert fine + SVC	0.498	0.418
ML baseline	0.418	0.338

Lecture:

- le rule-based reste référence absolue sur ce test,
- Camembert v2 est proche,
- la version v1 montre l'importance du volume et des epochs,
- les approches non fine-tunées restent nettement en retrait.

### 22.2 Niveau 2 - NLP sur gold manuel

Modele	Correct / Total	Accuracy
Rule-based	120 / 120	1.000
Camembert v2	119 / 120	0.992
ML baseline	67 / 120	0.558

Lecture:

- l'écart rule-based vs Camembert v2 est de 1 seul cas,
- l'écart vs baseline ML reste massif.

### 22.3 Niveau 3 - End-to-end

Backend NLP	NLP valides	Path valides (sur NLP valides)	Succès global
Rule-based	115/120	115/115	115/120
Camembert v2	115/120	115/115	115/120

Interpretation:

- sur ce lot, le pathfinding ne dégrade pas les cas valides,
- le verrou principal reste la couche NLP.

## **23. Retours d'experience: erreurs, ratés, pivots**

### **23.1 Raté 1 - Surconfiance dans un baseline neurale "plug-and-play"**

Hypothese initiale:

- utiliser Camembert sans fine-tuning devrait déjà dépasser un baseline classique.

Résultat:

- accuracy test ~0.498, insuffisant.

Apprentissage:

- les représentations générales ne suffisent pas si la tâche finale est spécifique et structurée.

### **23.2 Raté 2 - Croire qu'un score unique suffit**

Phase initiale:

- suivi prioritaire de l'accuracy globale.

Problème:

- cache des faiblesses INVALID et confusion origin/destination.

Correction:

- adoption d'un set métriques complet (`valid_f1`, `invalid_accuracy`, etc.).

### **23.3 Raté 3 - Relecture manuelle non priorisée**

Situation:

- lot manuel large, effort de correction diffus.

Problème:

- temps élevé pour gain incertain.

Pivot:

- feuille actionnable 22 IDs prioritaires.

Impact:

- accélération de la convergence du gold set.

### **23.4 Réussite 1 - Approche incrementaliste**

Choix:

- pipeline complet très tôt.

Impact:

- chaque itération produit des preuves évaluables,
- le projet avance même quand une branche expérimentale échoue.

## 23.5 Reussite 2 - Separation stricte NLP / pathfinding

Benefice:

- diagnostic propre des causes d'ehec,
- possibilite de comparer deux backends NLP a pathfinding constant.

## 23.6 Reussite 3 - Dashboard comparatif

Le dashboard `manual_gold_dashboard.json` est devenu la piece pivot:

- comparaison multi-modeles,
- lecture rapide en soutenance,
- base objective pour arbitrage final.

# 24. Couts, risques et compromis

## 24.1 Compromis performance vs maintenance

Rule-based:

- avantages: lisible, explicable, cout d'inference tres bas,
- limites: maintenance manuelle des regles et variantes.

Camembert v2:

- avantages: tres haute performance, potentiel de generalisation,
- limites: cout de train, dependance a torch/transformers, debugging moins intuitif.

## 24.2 Compromis court terme vs long terme

Court terme (livraison):

- rule-based est extremement solide, defendable immediatement.

Long terme (evolution):

- Camembert v2 ouvre une trajectoire de generalisation si le dataset manuel grossit.

Strategie recommandee:

- conserver les deux backends,
- choisir dynamiquement selon contexte (temps de reponse, confiance, perimetre).

## 24.3 Registre de risques

Risque	Probabilite	Impact	Mitigation
Distribution reelle differente du synthetique	Moyen	Eleve	augmenter set manuel, suivi d'erreurs terrain
Degradation silencieuse apres refactor	Moyen	Eleve	tests + snapshot avant merge
Cout compute Camembert	Faible/Moyen	Moyen	entrainement ponctuel, inference batchee
Dette documentaire	Moyen	Moyen	runbook + rapport long + bundle versionne

## **25. Protocole de soutenance technique**

### **25.1 Narratif conseille**

1. Probleme et contraintes.
2. Pourquoi un pipeline incremental.
3. Resultats comparatifs avec tableau unique.
4. Exemple E2E concret (une phrase, une sortie, un chemin).
5. Limites et feuille de route.

### **25.2 Questions jury anticipees**

Q: "Pourquoi ne pas livrer uniquement Camembert v2 ?" R: la double voie (**rule-based + camembert-ft**) maximise la robustesse et permet de garder un mode hautement explicable.

Q: "Pourquoi le ML baseline est-il faible ?" R: baseline volontairement simple pour reference scientifique; il permet de quantifier le gain des approches plus riches.

Q: "Comment prouvez-vous la reproductibilite ?" R: scripts eval et snapshot, commandes Make, bundle hashé, dashboard consolide.

## **26. Extension scientifique envisagee**

### **26.1 Experimentes non realises mais prioritaires**

- calibration de confiance des predictions Camembert,
- ensembling rule-based + Camembert selon score de confiance,
- passage a un pathfinding pondere temporellement,
- evaluation utilisateur sur phrases librement saisies (hors generateur).

### **26.2 Proposition de protocole post-projet**

1. constituer 500 a 1000 phrases manuelles hors templates,
2. annoter en double aveugle,
3. mesurer accord inter-annotateurs,
4. benchmarker de nouveau rule-based vs Camembert,
5. analyser les ecart par categorie d'erreur.

### **26.3 Critere de "production readiness"**

- accuracy NLP  $\geq 0.98$  sur jeu manuel etendu,
- invalid\_accuracy  $\geq 0.98$ ,
- taux d'echech pathfinding sur NLP valides  $< 1\%$ ,
- monitoring d'erreurs par type de phrase.

## **27. Conclusion et positionnement final**

Le projet demonstre qu'une demarche d'ingenierie pragmatique peut produire:

- des resultats de haut niveau,
- une traçabilite scientifique,
- une architecture evolutive,
- un discours de soutenance defendable.

Points saillants:

- baseline rule-based extremement performante,
- Camembert v2 quasi au niveau de la reference,
- pathfinding stable,
- instrumentation complete du cycle d'experience.

La vraie valeur n'est pas seulement le score final; c'est la capacite a expliquer pourquoi ce score existe, quelles decisions l'ont produit, et comment le systeme peut continuer a progresser de facon controlee.

---

## Annexe E - Catalogue d'erreurs observees (synthese)

### E.1 Erreurs NLP historiquement frequentes (avant stabilisation)

- inversion origin/destination sur formes telegraphiques,
- confusion destination/intermediaire apres "en passant par",
- faux positifs INVALID,
- bruit accent/typo sur noms composees.

### E.2 Erreurs residuelles apres stabilisation

- cas rares d'ambiguite contextuelle non levee,
- frontiere subtile entre requete d'information et requete de trajet.

### E.3 Actions correctives associees

- ajout de regles de cues,
- enrichissement des variantes de lieux,
- priorisation des relectures manuelles,
- bascule vers fine-tuning v2.

## Annexe F - Chronologie compacte du projet

1. Construction pipeline minimal et I/O conforme sujet.
2. Stabilisation rule-based + tests unitaires.
3. Ajout baseline ML et instrumentation metriques.
4. Integration GTFS, index gares, validation pathfinding.
5. Construction lot manuel + feuille de review priorisee.
6. Benchmarks spaCy / Camembert fige (resultats mitiges).
7. Fine-tune Camembert v1, puis v2 (gain majeur).
8. Integration backend multi-mode et dashboard comparatif.
9. Snapshot global et bundle de rendu hashé.

## 28. Details algorithmiques

Cette section formalise les principaux blocs pour renforcer le caractere scientifique du document.

### 28.1 Normalisation textuelle

Objectif: projeter des variantes orthographiques vers une representation stable.

Pipeline de normalisation:

1. passage en minuscule,
2. suppression des diacritiques,
3. suppression ponctuation/hors alphanumerique,
4. harmonisation des espaces et tirets.

Pseudo-code:

```
normalize(text):  
    text <- lower(text)  
    text <- remove_diacritics(text)  
    text <- replace_non_alnum_by_space(text)  
    text <- replace('-', ' ')  
    text <- collapse_spaces(text)  
    return trim(text)
```

Complexité:  $O(n)$  en taille de chaîne.

Impact pratique:

- réduit fortement la variance des formes,
- facilite les comparaisons exactes et approximatives,
- augmente la robustesse aux fautes simples.

## 28.2 Indexation des lieux

Un index par longueur de n-grammes est construit pour accélérer la recherche:

- bucket par nombre de tokens,
- sous-bucket par première lettre,
- fallback global si faute en première position.

Pseudo-code simplifié:

```
build_place_index(variants):  
    for (variant, canonical) in variants:  
        t <- tokens(variant)  
        L <- len(t)  
        index[L]['_all'].append((variant, canonical))  
        index[L][first_char(t[0])].append((variant, canonical))
```

Intérêt:

- évite un scan complet de toutes les variantes à chaque requête,
- conserve un fallback robuste pour typos "première lettre".

## 28.3 Matching fuzzy

Distance utilisée: Levenshtein avec prise en charge des transpositions courtes. Seuil dynamique `max_distance` selon longueur.

Règle de seuil:

- $\text{len} \leq 4: 0$
- $\text{len} \leq 6: 1$

- len <= 9: 2
- sinon: 3

Cette heuristique limite les faux positifs agressifs tout en conservant la correction des fautes usuelles.

#### 28.4 Extraction guidee par cues

L'extracteur cherche des patterns de type:

- depart cues (**de, depuis, ...**),
- destination cues (**a, vers, pour, ...**),
- gestion de gap tokens entre cue et lieu.

Pseudo-code:

```
extract_candidates(sentence_norm, cue_pattern, place_pattern, gap_max):
    regex <- cue_pattern + optional_gap(gap_max) + place_pattern
    for match in regex.finditer(sentence_norm):
        place <- canonicalize(match.place)
        add(position(match.place), place)
```

La fusion de candidats est ensuite ordonnee par position et desambiguisee.

#### 28.5 Pathfinding BFS

Le graphe est manipule en adjacency list.

Pseudo-code:

```
bfs(graph, source, target):
    queue <- [source]
    parent[source] <- None
    visited <- {source}
    while queue not empty:
        u <- pop_left(queue)
        if u == target: break
        for v in graph[u]:
            if v not in visited:
                visited.add(v)
                parent[v] <- u
                queue.push(v)
    return reconstruct_path(parent, source, target)
```

Complexite theorique:

- temps:  $O(V + E)$
- memoire:  $O(V)$

Dans notre cas, cette complexite est compatible avec la taille du graphe (3547 noeuds, 11430 aretes).

#### 28.6 Mapping ville -> stop area

Le module pathfinding realise une resolution progressive:

1. exact match,

2. prefix match,
3. fuzzy match,
4. variantes saint/st.

Ce design diminue les échecs en présence de bruit d'encodage et d'écriture non canonique.

## 29. Analyse de complexité et passage à l'échelle

### 29.1 Cote NLP rule-based

Facteurs de cout:

- normalisation  $O(n)$ ,
- scanning regex des cues,
- fuzzy local sur fenêtres candidates.

Borne pratique:

- pour des phrases courtes (< 30 tokens), le cout est faible,
- la latence per-request reste compatible CLI interactive.

### 29.2 Cote Camembert v2

Facteurs de cout:

- tokenization,
- forward pass transformeur,
- duplication origin/destination (deux modèles).

Observations:

- cout inférieur en batch qu'en singleton,
- choix batch-size impacte fortement le débit.

Compromis recommandé:

- inference unitaire pour demo,
- inference batchée pour évaluation massive.

### 29.3 Cote pathfinding

Le BFS sur  $V=3547$  et  $E=11430$  reste très abordable.

Le risque de performance n'est pas l'algorithme mais la résolution textuelle gares/IDs en cas de forte ambiguïté.  
D'où l'intérêt de conserver un index préconstruit.

## 30. Analyse de robustesse

### 30.1 Robustesse au bruit lexical

Le système absorbe:

- accents manquants,
- casse incohérente,
- fautes simples,
- variantes saint/st.

La combinaison normalisation + fuzzy est le principal levier rule-based.

### **30.2 Robustesse aux structures non canoniques**

Le module supporte:

- destination avant origine,
- prepositions alternatives,
- mots parasites.

Limite:

- phrases tres longues multi-intentions non segmentees restent plus fragiles.

### **30.3 Robustesse INVALID**

`invalid_accuracy` est un garde-fou central. Resultats forts:

- rule-based: 1.0 sur splits,
- Camembert v2: 1.0 sur dev/test et gold 120.

### **30.4 Robustesse end-to-end**

Le taux 115/115 de pathfinding sur NLP valides montre que le verrou de fiabilite est majoritairement en amont (NLP), et non dans la recherche de chemin.

## **31. Discussion sur la qualite des preuves**

### **31.1 Ce qui est solide**

- metriques multi-splits reproductibles,
- comparaison multi-modeles sur meme protocole,
- dashboard gold set consolidant rule-based/ML/Camembert,
- tests automatiques et bundle hashé.

### **31.2 Ce qui reste fragile**

- taille encore reduite du gold set manuel,
- possible proximite distributionnelle entre generateur et evaluation,
- absence de mesure d'accord inter-annotateurs formalisee.

### **31.3 Niveau de confiance**

Pour un projet academique de ce perimetre, le niveau de confiance est eleve sur:

- la validite interne du pipeline,
- la reproductibilite,
- la superiorite relative des modeles compares.

Il reste moyen sur la generalisation "terrain" sans campagne d'annotation supplementaire.

## **32. Gouvernance technique et qualite logicielle**

### **32.1 Hygiène de code**

- separation des scripts par responsabilite,
- CLI coherente,
- fichiers de metriques versionnes,
- tests unitaires couvrant les blocs critiques.

### **32.2 Traçabilité**

Chaque etape cle produit un artefact:

- metrique JSON,
- output CSV,
- resume markdown,
- bundle et manifest.

Ce design permet un audit ex-post des decisions.

### **32.3 Non regressions**

Le couple `make test + snapshot` joue le role de filet anti-regression. Toute evolution du pipeline est interpretable via les deltas de metriques.

## **33. Feuille de route vers version finale PDF**

### **33.1 Structure cible 20 pages**

Pour converger vers un format final de ~20 pages en style scientifique:

1. Page de garde + resume + abstract court.
2. Introduction et contexte.
3. Etat de l'art synthese (1-2 pages).
4. Donnees et protocole.
5. Architecture et details algorithmiques.
6. Resultats quantitatifs (tableaux).
7. Analyse qualitative (erreurs, cas).
8. Discussion, limites, perspectives.
9. Annexes techniques (commandes, artefacts, checks).

### **33.2 Elements visuels a inserer**

- Figure 1: architecture globale,
- Figure 2: flux donnees,
- Figure 3: evolution des performances par iteration,
- Figure 4: matrice comparative modeles,
- Tableau 1: metriques train/dev/test,
- Tableau 2: gold set,
- Tableau 3: E2E.

Ces visuels font naturellement gagner plusieurs pages tout en augmentant la lisibilite scientifique.

### **33.3 Texte deja pret**

Le present rapport contient deja:

- la narration technique,
- les metriques exactes,
- les details d'implementation,
- les erreurs et pivots,
- la partie reproductibilite.

La phase restante est principalement editoriale (mise en forme, figures, references bibliographiques).