

Travel Order Resolver

Rapport Scientifique - Version Jury

Ludovic Tagnon

2026-02-12

Table des matières

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

11.1	v1 rapide (preuve de concept)	9
11.2	v2 renforcee	10
11.3	Interpretation	10
12.	Pathfinding	10
12.1	Modele de graphe	10
12.2	Resolution des noms	11
12.3	Resultats	11
13.	Evaluation end-to-end	11
13.1	Lot manuel 120 (rule-based)	11
13.2	Lot manuel 120 (CamemBERT v2)	11
13.3	Gold manuel 120	11
14.	Analyse d'erreurs	12
14.1	Typologie des erreurs observees	12
14.2	Erreurs utiles (ratés)	12
14.3	Ce qui a marche	12
15.	Challenges techniques et strategie	12
15.1	Challenge 1: robustesse linguistique	12
15.2	Challenge 2: preuve de qualite	12
15.3	Challenge 3: maitrise du risque projet	12
16.	Discussion technique	13
16.1	Sur la performance	13
16.2	Sur la transferabilite	13
16.3	Sur le cout	13
17.	Validite et limites	13
17.1	Menaces de validite interne	13
17.2	Menaces de validite externe	13
17.3	Limites techniques actuelles	13
18.	Reproductibilite et artefacts	14
18.1	Scripts cles	14
18.2	Artefacts resultats	14
18.3	Bundle	14
19.	Etat actuel et plan avant soutenance	14
19.1	Etat des points critiques	14
19.2	Travail restant recommande	14
20.	Conclusion	14
22.	Comparaison quantitative multi-niveaux	15
22.1	Niveau 1 - NLP sur test synthetique	15
22.2	Niveau 2 - NLP sur gold manuel	15
22.3	Niveau 3 - End-to-end	15
23.	Retours d'experience: erreurs, ratés, pivots	16
23.1	Raté 1 - Surconfiance dans un baseline neurale "plug-and-play"	16
23.2	Raté 2 - Croire qu'un score unique suffit	16
23.3	Raté 3 - Relecture manuelle non priorisee	16
23.4	Reussite 1 - Approche incrementaliste	16
23.5	Reussite 2 - Separation stricte NLP / pathfinding	17
23.6	Reussite 3 - Dashboard comparatif	17
24.	Couts, risques et compromis	17
24.1	Compromis performance vs maintenance	17
24.2	Compromis court terme vs long terme	17
24.3	Registre de risques	17
25.	Protocole de soutenance technique	18
25.1	Narratif conseille	18
25.2	Questions jury anticipees	18
26.	Extensions envisagees	18

26.1 Experimentes non realises mais prioritaires	18
26.2 Proposition de protocole post-projet	18
26.3 Critere de "production readiness"	18
Annexe A - Commandes de reproduction	18
A.1 Verification globale	18
A.2 Baselines NLP	19
A.3 CamemBERT	19
A.4 Evaluation manuelle et dashboard	19
Annexe B - Tableau comparatif principal	19
Annexe C - Journal de decisions (synthese)	19
C.1 Decision 1	19
C.2 Decision 2	19
C.3 Decision 3	19
C.4 Decision 4	19
C.5 Decision 5	20

Page de garde

Projet: Travel Order Resolver

Formation: EPITECH

Auteur principal: Ludovic Tagnon

Version: Jury (compacte)

Date: 2026-02-12

\newpage

Sommaire visuel recommande

- Figure 1: Architecture globale (docs/figures/figure_1_architecture.png)
- Figure 2: Flux de donnees (docs/figures/figure_2_dataflow.png)
- Figure 3: Evolution metriques (docs/figures/figure_3_metrics_evolution.png)
- Figure 4: Taxonomie d'erreurs (docs/figures/figure_4_error_taxonomy.png)

\newpage

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 résultats, ce rapport documente la démarche, les erreurs, les impasses, les compromis techniques et les limites du protocole. L'objectif est de garder une trace claire pour la soutenance et pour la suite du 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 départ et une destination, puis de calculer un trajet ferroviaire cohérent à partir de données SNCF/GTFS. Le projet est volontairement transverse: NLP, qualité de données, algorithmique de graphe, reproductibilité expérimentale, et industrialisation.

Dans un cadre académique, le risque principal est de concentrer l'effort sur un modèle "avancé" sans stabiliser les fondamentaux (I/O, robustesse, protocole d'évaluation, outillage reproductible). La stratégie adoptée ici est l'inverse: stabiliser rapidement un pipeline mesurable puis complexifier par paliers.

1.2 Objectif technique

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

"Quel compromis entre robustesse, temps de dev, coût de calcul et qualité prédictive est le plus réaliste dans les contraintes du projet ?"

Cette question est traitée par comparaison expérimentale de plusieurs familles d'approches:

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

2. Problématique et contraintes

2.1 Formulation du problème

Entrée: une ligne `id,phrase`.

Sortie NLP:

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

Sortie pathfinding:

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

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

- fautes de frappe,
- absence d'accents,
- prépositions ambiguës,
- présence de villes intermédiaires,
- mots qui peuvent être prénoms 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 utile en pratique ?
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` (default),
- `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/submission_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

- entraînement 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'edition 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-guidee,
- forte redondance des patterns.

Une logique deterministe bien calibre capture efficacement ce regime.

8.3 Resultats

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 generalisation train->dev->test,
- stabilite 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 ete introduit pour remplacer le rule-based immediatement, mais pour fournir:

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

9.2 Resultats

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 modele sur-apprend partiellement le train et generalise mal. Il confond frequemment:

- origine/destination inversees,
- villes intermediaires prises pour destination,
- faux positifs sur phrases ambiguës.

Exemples d'erreurs:

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

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 depasse clairement le 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 confirme que la valeur venait du fine-tuning, pas du simple usage d'embeddings pre-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 separees (origin/destination).

Resultats 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 entrainement:

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

Resultats 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 resultat montre que l'approche neurale est performante une fois:

- le volume de train augmente,
- les hyperparametres sont stabilises,
- le mode end-to-end est assume.

12. Pathfinding

12.1 Modele de graphe

Le reseau est represente par un graphe orienté des **StopArea**. La recherche de chemin est traitee par un algo de plus court chemin de type BFS (non pondere).

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 (prenoms, 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:

- multiplicite de metriques,
- evaluation multi-splits,
- dashboard comparatif,
- lot manuel gold.

15.3 Challenge 3: maitrise du risque projet

Probleme: une approche purement "modele avance" etait risquee en delai.

Reponse strategique:

- baseline rule-based d'abord,

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

16. Discussion technique

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-contraint, 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'ou 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 entraînement/inference plus eleve.

17. Valide et limites

17.1 Menaces de valide 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 valide 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 cles

- 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 resultats

- `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: finalise.

19.2 Travail restant recommande

1. augmenter le lot manuel annotate (objectif > 300 phrases),
2. verifier les ecart rule-based vs CamemBERT sur des cas reels hors distribution,
3. eventuellement ajouter un ranking confiance pour la sortie NLP,
4. produire la version PDF finale avec schema d'architecture et tableaux numerotes.

20. Conclusion

Le projet atteint un niveau de maturite technique correct:

- solution operationnelle complete NLP + pathfinding,
- niveau de performance eleve et mesure,
- documentation reproductible,
- preuves quantitatives et qualitatives sur plusieurs familles de modeles.

La contribution principale est moins "un modele" qu'une architecture et une demarche:

- 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 techniquement, audite, reproduit, et prolonge.

22. Comparaison quantitative multi-niveaux

22.1 Niveau 1 - NLP sur test synthetique

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 fige + SVC	0.498	0.418
ML baseline	0.418	0.338

Lecture:

- le rule-based reste reference absolue sur ce test,
- Camembert v2 est proche,
- la version v1 montre l'importance du volume et des epochs,
- les approches non fine-tunees 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'ecart rule-based vs Camembert v2 est de 1 seul cas,
- l'ecart vs baseline ML reste massif.

22.3 Niveau 3 - End-to-end

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

Interpretation:

- sur ce lot, le pathfinding ne degrade 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 deja dépasser un baseline classique.

Resultat:

- accuracy test ~0.498, insuffisant.

Apprentissage:

- les representations generales ne suffisent pas si la tache finale est specifique et structuree.

23.2 Raté 2 - Croire qu'un score unique suffit

Phase initiale:

- suivi prioritaire de l'accuracy globale.

Probleme:

- cache des faiblesses INVALID et confusion origin/destination.

Correction:

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

23.3 Raté 3 - Relecture manuelle non priorisee

Situation:

- lot manuel large, effort de correction diffus.

Probleme:

- temps eleve pour gain incertain.

Pivot:

- feuille actionnable 22 IDs prioritaires.

Impact:

- acceleration de la convergence du gold set.

23.4 Reussite 1 - Approche incrementaliste

Choix:

- pipeline complet tres tot.

Impact:

- chaque iteration produit des preuves evaluables,
- le projet avance meme quand une branche experimentale echoue.

23.5 Reussite 2 - Separation stricte NLP / pathfinding

Benefice:

- diagnostic propre des causes d'echec,
- 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. Coûts, 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, exploitable 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
Dettes documentaires	Moyen	Moyen	runbook + rapport long + bundle versionne

25. Protocole de soutenance technique

25.1 Narratif conseil

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; il sert de point de comparaison pour 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. Extensions envisagees

26.1 Experimentes non realisees 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 ≥ 0.98 sur jeu manuel etendu,
- `invalid_accuracy` ≥ 0.98 ,
- taux d'echec pathfinding sur NLP valides $< 1\%$,
- monitoring d'erreurs par type de phrase.

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: point de comparaison simple 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.