

الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي



جامعة سعيدة د. مولاي الطاهر

كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية و اللاسلكية

قسم: الإعلام الآلي

Mémoire de Master en informatique Spécialité : intelligence artificielle principe et application

T h è m e

Conception et mise en œuvre d'un système intelligent d'aide au triage médical fondé sur un modèle de langage médical spécialisé

Cas du système BetterEMS

■ Présenté par :

Mme Boutaleb Imene

■ Dirigé par

MR Rahemani Mohamed Elhadi

Année universitaire



2025-2026

Remerciements

Au terme de ce travail, je tiens à exprimer ma reconnaissance à toutes les personnes qui en ont rendu la réalisation possible.

Je remercie en premier lieu mon encadreur, Dr. Rahmani Mohamed Elhadi, pour la confiance qu'il m'a accordée, la justesse de ses orientations et la régularité de son suivi. Ses remarques ont, à plusieurs reprises, redressé la trajectoire de ce projet.

Ma gratitude va également à M. Merzoug Rayane, partenaire du projet, dont l'appui sur la mise en production et l'infrastructure a permis de faire passer BetterEMS du prototype au système réellement déployé.

Je remercie le Centre de Recherche en Informatique (CRI) de l'Université de Saïda, et particulièrement M. Baki Abdtouab, pour l'accès aux moyens de calcul sans lesquels l'entraînement du modèle n'aurait pas été possible.

J'adresse mes remerciements aux membres du jury qui ont accepté d'évaluer ce travail, ainsi qu'à l'ensemble des enseignants du Département d'Informatique pour la formation reçue durant ces années.

Enfin, je remercie mon mari pour le soutien le long de mon expérience de ma formation mes enfants ma famille et mes proches pour leur soutien constant.

Dédicaces

À mon marie mes enfants Anes et zakaria, pour leur patience et leurs encouragements sans relâche.

À mes parents mon frère et mes sœurs,a

Rayane merzougue pour leur présence. À mes amis et à mes camarades de

promotion, compagnons de ce parcours. À toutes celles et ceux qui ont

contribué, de près ou de loin, à ce travail.

Déclaration sur l'honneur

Je soussignée, **Boutaleb Imane**, déclare sur l'honneur que ce mémoire, intitulé « Conception et mise en œuvre d'un système intelligent d'aide au triage médical fondé sur un modèle de langage médical spécialisé : Cas du système BetterEMS », est le fruit de mon propre travail, réalisé sous la direction de Dr. Rahmani Mohamed Elhadi.

Je certifie que ce travail n'a jamais été soumis auparavant en vue de l'obtention d'un autre diplôme, et que toutes les sources utilisées ont été citées et référencées conformément aux règles en vigueur. Tout emprunt à des travaux tiers (textes, figures, données ou code) y est explicitement signalé.

Je suis consciente que tout plagiat ou toute fraude avérée entraînerait les sanctions prévues par le règlement de l'Université de Saïda, Dr. Moulay Tahar.

Fait à Saïda, le

Signature de l'étudiante

Table des matières

Remerciements	i
Dédicaces	ii
Déclaration sur l'honneur	iii
Table des figures	iv
Liste des tableaux	v
Liste des abréviations	vi
Introduction générale	1
1 Files d'attente et triage médical	4
1.1 Introduction	4
1.2 Théorie des files d'attente	5
1.2.1 Composantes d'un système de file d'attente	5
1.2.2 Notation de Kendall et modèles M/M/1, M/M/c	6
1.2.3 Taux d'occupation du système	6
1.2.4 Loi de Little	7
1.2.5 Limites des modèles classiques	8
1.3 Files d'attente prioritaires	8
1.3.1 La discipline FIFO et ses limites aux urgences	8
1.3.2 Priorité préemptive et non préemptive	9
1.3.3 Effet des priorités sur les temps d'attente	10
1.4 Le triage médical aux urgences	11
1.4.1 Définition et origines	11
1.4.2 Enjeux cliniques	12
1.4.3 Déroulement du triage	12
1.4.4 Erreurs de triage : sous-triage et sur-triage	13
1.4.5 Limites du triage manuel	13
1.5 Les systèmes de triage normalisés	14
1.5.1 Manchester Triage System	14
1.5.2 Emergency Severity Index	15

1.5.3	Canadian Triage and Acuity Scale.....	16
1.5.4	Comparaison et choix pour BetterEMS	17
1.6	Les urgences hospitalières en Algérie	18
1.7	Conclusion	19
2	État de l'art : de l'apprentissage automatique aux modèles de langage médicaux	20
2.1	Introduction	20
2.2	Apprentissage automatique	20
2.2.1	Définition et paradigmes	20
2.2.2	Concepts fondamentaux.....	21
2.2.3	Surapprentissage et sous-apprentissage	22
2.2.4	Métriques d'évaluation	22
2.3	Apprentissage profond	23
2.3.1	Le neurone artificiel et le perceptron.....	24
2.3.2	Réseaux multicouches.....	24
2.3.3	Fonctions d'activation.....	25
2.3.4	Rétropropagation et descente de gradient	26
2.3.5	GPU et accélération matérielle	27
2.4	Traitement automatique du langage naturel.....	27
2.4.1	Représenter le texte : du sac de mots aux embeddings	28
2.4.2	L'architecture Transformer.....	28
2.4.3	Familles de modèles : encodeurs et décodeurs	29
2.4.4	Tokenisation et embeddings	30
2.5	Grands modèles de langage	31
2.5.1	Pré-entraînement et capacités émergentes.....	31
2.5.2	Du modèle généraliste au modèle spécialisé	31
2.5.3	Quantification : fp16, INT8, INT4 et GGUF.....	32
2.5.4	Inférence locale ou cloud	33
2.5.5	LLMs médicaux	33
2.6	Adapter un LLM : le fine-tuning efficace.....	34
2.6.1	Transfer learning et PEFT	34
2.6.2	LoRA	35
2.6.3	QLoRA et ses arbitrages	36
2.6.4	Prompt engineering et sorties structurées	36
2.7	L'IA appliquée au triage des urgences	37
2.7.1	Panorama des travaux	37
2.7.2	Tableau comparatif	38
2.7.3	Lacunes et positionnement	39
2.8	Conclusion	40

3	Conception et mise en œuvre du système BetterEMS	41
3.1	Introduction	41
3.2	Analyse des besoins et acteurs.....	41
3.3	Architecture globale.....	44
3.4	Architecture de déploiement.....	47
3.5	Constitution du corpus propriétaire.....	48
3.5.1	Création manuelle des cas.....	48
3.5.2	Nettoyage et contrôle qualité.....	49
3.5.3	Découpage et distribution du corpus	49
3.5.4	Format des exemples d'entraînement.....	50
3.6	Notre modèle : BetterEMS-Triage	51
3.6.1	Choix du modèle de base	51
3.6.2	Prompt système et sortie structurée	51
3.6.3	Stratégie de fine-tuning.....	52
3.6.4	Export et quantification	53
3.7	Intégration de l'IA dans le système	54
3.7.1	Service d'inférence	54
3.7.2	Mécanismes de sécurité	54
3.7.3	File d'attente temps réel.....	55
3.7.4	Gestion des rôles et des accès.....	56
3.8	Apprentissage continu.....	56
3.9	Conclusion	57
4	Expérimentations et résultats	58
4.1	Environnement expérimental.....	58
4.2	Démarche expérimentale : deux stratégies de fine-tuning comparées.....	59
4.3	Entraînement de BetterEMS-Triage (LoRA fp16).....	60
4.3.1	Configuration retenue	60
4.3.2	Courbes d'apprentissage	61
4.3.3	Portée de la mesure.....	64
4.4	Performances d'inférence.....	64
4.5	Comparaison avec l'algorithme à règles	65
4.5.1	Évaluation chiffrée de l'algorithme de repli.....	66
4.6	Validation fonctionnelle de bout en bout.....	68
4.7	Bilan et perspectives	68
4.8	Conclusion	69
5	Étude entrepreneuriale : de la recherche à la startup	71
5.1	Introduction.....	71
5.2	Résumé du projet et équipe	71

Table des matières	Aide au triage médical par IA
5.3 Problématique ciblée.....	73
5.4 Solution proposée et prototype.....	74
5.5 Valeur ajoutée.....	75
5.6 Analyse du marché cible	76
5.7 Analyse concurrentielle	76
5.8 Modèle économique.....	79
5.9 Feuille de route	81
5.10 Lean Canvas	83
5.11 Conclusion	84
Conclusion générale	85
Bibliographie	88
A Exemples de cas du jeu de données	91
B Prompt système de triage (extrait)	94
C Configuration du fine-tuning (extrait)	96
D Intégration du modèle dans le backend (extrait)	98
E Configuration de déploiement (extrait)	100
F Captures d'écran de l'application	101
G Glossaire	106
H Résumé	107

Table des figures

1.1	Structure générale d'un système de file d'attente.	9
1.2	Discipline FIFO et file à priorité : positionnement d'un cas critique.	9
1.3	Processus de triage aux urgences (de l'admission à l'orientation).	12
1.4	Les cinq niveaux du Manchester Triage System et leurs délais cibles.	15
2.1	Réseau de neurones multicouches (8-12-10-6-5) pour une classification à cinq classes.	24
2.2	Fonctions d'activation sigmoïde et ReLU	26
2.3	Architecture Transformer (vue simplifiée).	29
2.4	Principe de LoRA : matrices basse-rang A et B en parallèle du poids gelé W.	35
3.1	Diagramme de cas d'utilisation de BetterEMS.	43
3.2	Diagramme de séquence du processus de triage.	44
3.3	Architecture globale de BetterEMS.	45
3.4	Diagramme de classes du modèle de données.	46
3.5	Architecture de déploiement : VPS applicatif et serveur GPU reliés par un réseau privé virtuel sécurisé.	47
3.6	Constitution du corpus de triage (12 600 cas rédigés à la main, découpés en 12 400 / 100 / 100).	49
3.7	Distribution des classes de triage dans le corpus (n = 12 400).	50
4.1	Courbes de perte d'entraînement et de validation (3 époques, données réelles).	62
4.2	Token accuracy au cours de l'entraînement (données réelles).	62
4.3	Matrice de confusion de l'algorithme de repli sur les 100 cas de test (lignes : niveau réel ; colonnes : niveau prédit).	67
5.1	Compte de résultat prévisionnel sur trois ans (modèle B2G récurrent).	80
5.2	Diagramme de Gantt : planning de réalisation du projet.	83
5.3	Lean Canvas du projet BetterEMS.	83
F.1	Page de connexion : authentification du personnel par rôle.	101
F.2	Tableau de bord : vue d'ensemble du service (volumes, niveaux de gravité, temps d'attente).	101
F.3	Enregistrement d'un patient par l'agent d'accueil (étape 1 sur 5).	102
F.4	Enregistrement d'un patient (étape 2 sur 5).	102
F.5	Enregistrement d'un patient (étape 3 sur 5).	103

F.6	Écran de triage : patients triés par niveau d'urgence, avec couleur Manchester, score de confiance et indicateurs d'alerte produits par le modèle.....	103
F.7	File d'attente priorisée, mise à jour en temps réel.	104
F.8	Écran d'appel de la salle d'attente (affichage TV) : numéro de ticket, box et couleur du patient appelé.	104
F.9	Console d'administration du moteur d'intelligence artificielle (état du service, disponibilité, repli).....	105
F.10	Gestion des utilisateurs et des rôles (administrateur).	105

Liste des tableaux

1.1	Délais cibles de prise en charge par niveau de priorité (valeurs inspirées du Manchester Triage System [6]).	11
1.2	Les cinq niveaux du Manchester Triage System [6].	15
1.3	Les cinq niveaux de l'Emergency Severity Index [9].	16
1.4	Les cinq niveaux de la Canadian Triage and Acuity Scale [8].	17
1.5	Comparaison des trois systèmes de triage normalisés [6], [8], [9].	17
2.1	Matrice de confusion d'un classifieur binaire.	23
2.2	Décompte des paramètres entraînaables du réseau 8-12-10-6-5.	25
2.3	Deux familles de modèles issues du Transformer.	30
2.4	Formats de précision numérique et empreinte mémoire approximative pour un modèle de 7 milliards de paramètres.	32
2.5	LoRA et QLoRA : termes de l'arbitrage entre mémoire et stabilité.	36
2.6	Travaux représentatifs sur le triage assisté par IA et positionnement de BetterEMS.	38
3.1	Les cinq rôles du système BetterEMS et leurs responsabilités.	42
3.2	Les neuf entités du modèle de données de BetterEMS.	45
3.3	Hyperparamètres de fine-tuning de BetterEMS-Triage (configuration retenue).	53
4.1	Environnement matériel et logiciel de la campagne expérimentale.	59
4.2	Configuration QLoRA testée (hypothèse 1) et issue des trois essais.	60
4.3	Hyperparamètres de l'entraînement de BetterEMS-Triage.	61
4.4	Pertes et précision par époque; le meilleur point de validation est atteint à l'époque 1.	63
4.5	Comparaison qualitative entre l'algorithme à règles et notre modèle sur les cas de test.	65
4.6	Performance par niveau de l'algorithme de repli sur l'ensemble de test (n = 100).	67
5.1	Équipe du projet BetterEMS et répartition des rôles.	72
5.2	Problèmes identifiés chez le client cible, avec leur impact et leur fréquence.	73
5.3	Concurrents directs et indirects de BetterEMS.	77
5.4	Analyse SWOT du projet BetterEMS.	78
5.5	Compte de résultat prévisionnel sur trois ans (DA).	80
5.6	Feuille de route du projet sur trois ans.	82

Liste des abréviations

Sigle	Signification
AdamW	<i>Adam with decoupled Weight decay</i> (algorithme d'optimisation)
API	<i>Application Programming Interface</i> (interface de programmation)
AVC	Accident vasculaire cérébral
B2G	<i>Business-to-Government</i> (vente au secteur public)
BERT	<i>Bidirectional Encoder Representations from Transformers</i>
bf16	<i>Brain Floating Point</i> sur 16 bits
BPE	<i>Byte Pair Encoding</i> (tokenisation en sous-mots)
CAC	Coût d'Acquisition Client
CHU	Centre Hospitalo-Universitaire
CIM-10	Classification internationale des maladies, 10 ^e révision
CPU	<i>Central Processing Unit</i> (processeur central)
CRI	Centre de Recherche en Informatique (Université de Saïda)
CTAS	<i>Canadian Triage and Acuity Scale</i>
CUDA	<i>Compute Unified Device Architecture</i> (calcul GPU NVIDIA)
DA	Dinar algérien (monnaie)
ESI	<i>Emergency Severity Index</i>
EPH	Établissement Public Hospitalier
F1	Moyenne harmonique de la précision et du rappel
FIFO	<i>First In, First Out</i> (premier arrivé, premier servi)
fp16	Virgule flottante sur 16 bits (demi-précision)
GCS	<i>Glasgow Coma Scale</i> (échelle de coma de Glasgow)
GGUF	<i>GPT-Generated Unified Format</i> (format de modèle quantifié)
GPU	<i>Graphics Processing Unit</i> (processeur graphique)
IA	Intelligence Artificielle
JSON	<i>JavaScript Object Notation</i>
JWT	<i>JSON Web Token</i>
LLM	<i>Large Language Model</i> (grand modèle de langage)
LoRA	<i>Low-Rank Adaptation</i>
LTV	<i>Lifetime Value</i> (valeur vie client)
MLP	<i>Multi-Layer Perceptron</i> (perceptron multicouche)
MTS	<i>Manchester Triage System</i>
NF4	<i>4-bit NormalFloat</i> (quantification sur 4 bits)

Sigle	Signification
NLP	<i>Natural Language Processing</i> (traitement du langage naturel)
ORM	<i>Object-Relational Mapping</i> (mapping objet-relationnel)
PEFT	<i>Parameter-Efficient Fine-Tuning</i>
PFE	Projet de Fin d'Études
Q4_K_M	Schéma de quantification sur 4 bits de llama.cpp
QLoRA	<i>Quantized Low-Rank Adaptation</i>
RBAC	<i>Role-Based Access Control</i> (contrôle d'accès par rôles)
ReLU	<i>Rectified Linear Unit</i>
REST	<i>Representational State Transfer</i>
SAMU	Service d'Aide Médicale Urgente
SIH	Système d'Information Hospitalier
SWOT	<i>Strengths, Weaknesses, Opportunities, Threats</i> (forces, faiblesses, opportunités, menaces)
TALN	Traitement Automatique du Langage Naturel
TLS	<i>Transport Layer Security</i> (chiffrement des communications)
UML	<i>Unified Modeling Language</i> (langage de modélisation)
VPN	<i>Virtual Private Network</i> (réseau privé virtuel)
VRAM	<i>Video Random Access Memory</i> (mémoire vive du GPU)

Introduction générale

Quiconque a passé une soirée d'affluence dans un service d'urgences algérien connaît la scène : une salle d'attente saturée, des soignants qui passent d'un box à l'autre sans relâche, et des patients appelés, le plus souvent, dans l'ordre où ils se sont présentés au guichet. Cet ordre d'arrivée ne dit pourtant rien de l'état clinique. Une douleur thoracique d'origine coronarienne peut franchir la porte après une dizaine de consultations bénignes ; tant que la file reste strictement chronologique, c'est elle qui patiente. Pour ce patient-là, chaque minute d'attente a un coût clinique direct, parfois irréversible.

Le problème n'est pas propre à l'Algérie, mais il s'y pose avec une intensité que connaissent bien les équipes de garde. La demande de soins urgents croît d'année en année, tandis que les effectifs et les locaux suivent difficilement. Surtout, le tri à l'admission, lorsqu'il existe, repose sur l'appréciation de l'infirmier d'accueil, sans protocole appliqué de manière uniforme. Deux soignants peuvent classer différemment le même patient ; le même soignant, en fin de garde, ne classe plus tout à fait comme il le faisait huit heures plus tôt. Cette variabilité du triage manuel est documentée de longue date dans la littérature [10], et elle s'aggrave précisément aux heures où le tri serait le plus nécessaire, quand l'affluence et la fatigue se cumulent.

Des protocoles standardisés existent pourtant. Le Manchester Triage System, le Canadian Triage and Acuity Scale ou l'Emergency Severity Index classent chaque patient sur cinq niveaux d'urgence assortis de délais cibles de prise en charge. Le protocole de Manchester, que nous avons retenu comme référence, associe à chaque niveau une couleur et une attente maximale : rouge pour la prise en charge immédiate, orange pour dix minutes, jaune pour une heure, vert pour deux heures, bleu pour quatre heures [6]. Ces grilles ont fait la preuve de leur fiabilité [7], mais leur application exige du personnel formé, du temps à l'admission et une rigueur constante. C'est exactement ce qui fait défaut aux heures de pointe, et c'est là que réside l'écart entre le protocole sur le papier et la pratique au guichet.

L'intelligence artificielle offre une piste sérieuse pour combler cet écart. Plusieurs équipes ont montré qu'un modèle entraîné sur des données d'admission pouvait prédire le niveau d'urgence avec une fiabilité comparable, parfois supérieure, à celle du triage humain [29], [32]. Les modèles de langage de grande taille élargissent encore le champ : capables de lire un motif de consultation rédigé en langage naturel, ils croisent symptômes, antécédents et constantes vitales sans imposer de formulaire rigide, et restituent un raisonnement que le soignant peut lire et contester [36]. Des déclinaisons médicales ouvertes, comme BioMistral-7B, pré-entraîné sur la littérature biomédicale et capable de raisonner en français, rendent cette approche accessible en dehors des grands laboratoires [23].

Rien de tout cela ne se transpose cependant tel quel dans un hôpital algérien. Les solutions commerciales de triage assisté sont rares, coûteuses, rarement disponibles en français, et presque

toujours hébergées sur des clouds étrangers, ce qui est difficilement acceptable pour des données de santé. La question qui structure ce mémoire peut donc se formuler simplement : peut-on construire, avec des moyens locaux, un système d'aide au triage fondé sur un modèle de langage médical, assez rapide pour s'insérer dans le flux réel d'un service d'urgences, assez souverain pour que les données des patients ne quittent jamais l'infrastructure de soins, et conçu dès l'origine pour que le médecin garde le dernier mot ?

Notre réponse est le système BetterEMS, dont ce mémoire retrace la conception, la réalisation et l'évaluation. Quatre objectifs en ont guidé le développement. Le premier est d'adapter un modèle de langage médical au triage selon le protocole de Manchester, par fine-tuning sur des cas rédigés en français et représentatifs des pathologies rencontrées en Algérie. Le deuxième consiste à intégrer ce modèle dans une chaîne complète, de l'enregistrement administratif du patient à son appel en box de consultation, avec une file d'attente priorisée et mise à jour en temps réel. Le troisième fait de la validation humaine une étape obligatoire du circuit : chaque proposition du modèle est confirmée ou corrigée par un médecin avant de prendre effet, et le système conserve la trace de chaque décision. Le dernier objectif transforme ces corrections en matière première d'un apprentissage continu, afin que le modèle s'améliore à mesure qu'il est utilisé, au plus près des pratiques locales.

Les contributions de ce travail sont concrètes et vérifiables. Nous avons d'abord constitué un jeu de données de triage en français adapté au contexte algérien : 12 400 cas cliniques rédigés manuellement, couvrant la cardiologie, la traumatologie, la pédiatrie, les intoxications ou encore les pathologies infectieuses fréquentes sur le terrain local, nettoyés et validés pour constituer un corpus propriétaire de triage, chacun annoté selon les cinq niveaux de Manchester et associé à des codes CIM-10 puisés dans un dictionnaire d'environ 109 000 entrées ; le modèle présenté dans ce mémoire a été entraîné sur ce corpus de 12 400 exemples. Nous avons ensuite construit notre modèle BetterEMS-Triage par adaptation LoRA en précision fp16 [24] du modèle biomédical ouvert retenu : 7,24 milliards de paramètres, dont environ 84 millions ajustés au cours d'un entraînement mené sur le serveur GPU du Centre de Recherche en Informatique (CRI) de l'Université de Saïda ; le modèle atteint ≈ 96 % de précision (token accuracy) sur l'ensemble de validation. Cette configuration est le fruit d'une démarche expérimentale maîtrisée : notre premier essai, en QLoRA quantifié sur 4 bits, a divergé dès le début de l'entraînement malgré plusieurs réglages du taux d'apprentissage ; le diagnostic de cette instabilité numérique, puis le passage à un entraînement en fp16 qui l'a résolue, constituent à nos yeux un enseignement méthodologique à part entière. Autour du modèle, quantifié pour rendre sa proposition en deux à quatre secondes par patient, nous avons développé un système web complet : cinq rôles d'utilisateurs, neuf entités métier, une file d'attente en temps réel, du backend à l'écran d'appel de la salle d'attente, le tout déployé en production sur une infrastructure associant un serveur applicatif et le serveur GPU du CRI. Enfin, le projet a dépassé le cadre du mémoire : il a été retenu comme startup labellisée au titre du décret ministériel 1275, avec un brevet d'invention en cours de dépôt, ce qui donne à ce travail un double caractère, académique et entrepreneurial.

Une précaution mérite d’être posée dès maintenant, parce qu’elle a orienté chaque choix de conception. BetterEMS est un outil d’aide à la décision. Le niveau de triage qu’il propose reste une proposition, affichée avec son indice de confiance et son explication ; la décision appartient au médecin, qui la confirme ou la corrige en quelques secondes. Cette retenue n’est pas une clause de style : elle correspond aux recommandations internationales sur l’usage de l’intelligence artificielle en santé [30], et elle conditionne l’acceptabilité du système par les équipes soignantes.

Le mémoire s’organise en cinq chapitres. Le premier pose les fondements du problème : théorie des files d’attente, limites de la discipline FIFO en milieu hospitalier, présentation des protocoles de triage et du contexte algérien. Le deuxième dresse un état de l’art qui va de l’apprentissage automatique classique aux modèles de langage médicaux, en passant par les architectures Transformer et les techniques d’adaptation à faible coût telles que LoRA. Le troisième détaille la conception et la mise en œuvre de BetterEMS : architecture logicielle, constitution du jeu de données, stratégie de fine-tuning, mécanismes de sécurité et déploiement. Le quatrième rapporte les expérimentations et leurs résultats, de la première tentative QLoRA à la configuration finale, et se clôt sur les perspectives de la feuille de route. Le cinquième, enfin, présente l’étude entrepreneuriale du projet, du prototype à la startup, conformément au canevas officiel des projets de fin d’études à caractère innovant. Une conclusion générale rassemble les enseignements du travail et trace les étapes planifiées qui mèneront le système déployé jusqu’à la routine hospitalière.

Files d'attente et triage médical

1.1 Introduction

L'attente naît partout où une demande variable rencontre une capacité de service limitée. Le phénomène est si général qu'il a donné naissance à une branche entière de la recherche opérationnelle : la théorie des files d'attente, qui étudie depuis les premiers travaux sur les centraux téléphoniques la manière dont des clients arrivent, patientent et sont servis [1]. Guichets administratifs, routeurs de réseaux, chaînes de production, centres d'appels : tous se laissent décrire avec le même vocabulaire, celui d'un flux d'arrivées, d'un ou plusieurs serveurs, et d'une règle qui décide qui passe en premier.

Dans la plupart de ces systèmes, la règle naturelle est l'ordre d'arrivée. Elle est simple, transparente, perçue comme équitable puisqu'elle ne demande aucune information sur les clients. Les urgences hospitalières font exception, et cette exception est tout le sujet de ce chapitre. Un service d'urgences reçoit côte à côte des situations bénignes et des détresses vitales, sans aucune corrélation entre l'heure d'arrivée et la gravité : le patient le plus grave de la journée peut très bien franchir la porte derrière trois cas mineurs. Appliquer l'ordre chronologique revient alors à faire attendre celui qui ne le peut pas.

La réponse des hôpitaux à ce problème porte un nom : le triage. Classer chaque patient dès son admission selon la gravité de son état, puis organiser la file d'attente autour de ce classement plutôt qu'autour de l'horloge : voilà, en deux phrases, le principe qui structure les services d'urgences modernes [6]. Derrière sa simplicité apparente se cachent une théorie mathématique bien établie, celle des files à priorité, et une pratique clinique exigeante, codifiée par plusieurs protocoles internationaux.

Ce chapitre construit le socle conceptuel sur lequel repose le système BetterEMS. Nous partons des notions élémentaires de la théorie des files d'attente (section 1.2), puis des files à priorité qui en corrigent la principale faiblesse (section 1.3). Nous présentons ensuite le triage médical proprement dit, son déroulement et ses erreurs caractéristiques (section 1.4), avant de comparer les trois grands systèmes normalisés (Manchester, ESI et CTAS) et de justifier le choix du protocole de Manchester comme référence de notre travail (section 1.5). Un état des lieux des urgences hospitalières algériennes (section 1.6) précise enfin le contexte auquel ce travail est destiné.

1.2 Théorie des files d'attente

La théorie des files d'attente est née au début du XX^e siècle des travaux d'Agner Krarup Erlang sur le dimensionnement des centraux téléphoniques, avant d'être systématisée comme un chapitre à part entière des probabilités appliquées [1]. Son objet est de prédire le comportement d'un système de service à partir de quelques grandeurs mesurables : à quel rythme les clients arrivent-ils, combien de temps dure un service, combien de serveurs sont disponibles. De ces données, elle déduit des indicateurs de performance, comme le temps moyen d'attente, la longueur de la file ou le taux d'occupation, qui permettent de dimensionner un système ou d'en diagnostiquer les dérives [2].

L'hôpital se prête naturellement à cette lecture. Les patients sont les clients ; les médecins, les infirmiers et les boxes d'examen sont les serveurs ; la salle d'attente matérialise la file. Avant d'examiner ce que cette analogie a de trompeur, car elle l'est sur un point décisif, il faut en poser les éléments.

1.2.1 Composantes d'un système de file d'attente

Quatre éléments suffisent à caractériser un système de file d'attente élémentaire [2]. Nous les présentons ici en donnant chaque fois leur lecture hospitalière.

Processus d'arrivée. Il décrit la manière dont les clients entrent dans le système, résumée par le débit moyen λ : le nombre moyen d'arrivées par unité de temps. Les arrivées réelles sont rarement régulières ; le modèle classique les représente par un processus de Poisson, dans lequel les événements surviennent indépendamment les uns des autres et sans mémoire, si bien que le fait qu'un client vienne d'arriver ne renseigne en rien sur l'instant du suivant [1]. Aux urgences, λ se lit en patients par heure. Il varie fortement selon l'heure, le jour et la saison ; l'hypothèse d'un débit constant n'est donc défendable que sur des plages horaires courtes.

Processus de service. Symétriquement, le taux μ mesure la capacité de traitement d'un serveur : le nombre moyen de clients qu'il peut servir par unité de temps, la durée moyenne d'un service valant $1/\mu$. L'hypothèse classique est celle d'une durée exponentielle : beaucoup de services courts, quelques services très longs [1]. La médecine pousse cette dispersion à l'extrême. Une suture simple se règle en quelques minutes ; un polytraumatisé mobilise une équipe entière pendant des heures. Cette variabilité des durées est l'une des sources principales de congestion des services d'urgences [10].

Discipline de service. C'est la règle qui désigne le prochain client à servir. La plus répandue est le FIFO (*First In, First Out*) : premier arrivé, premier servi. On rencontre aussi le LIFO, où le dernier arrivé passe d'abord, selon la logique des piles en informatique, le partage cyclique du temps (*round robin*) des ordonnanceurs de processus, et les disciplines à priorité, qui servent d'abord les clients jugés les plus importants. Un point mérite d'être souligné d'emblée : la discipline ne change rien à la capacité du système. Elle ne fait que répartir l'attente entre les

clients ; c'est elle qui décide qui paie le prix de la congestion [3].

Nombre de serveurs. Le système peut comporter un serveur unique ou c serveurs en parallèle alimentés par une même file. Un cabinet à médecin seul relève du premier cas ; un service d'urgences, avec ses boxes multiples et son équipe médicale, relève du second. À capacité totale égale, une file unique desservant plusieurs serveurs est d'ailleurs plus efficace que plusieurs files séparées : aucun serveur n'y reste inactif pendant qu'une file voisine déborde [2].

1.2.2 Notation de Kendall et modèles M/M/1, M/M/c

Pour désigner ces systèmes sans ambiguïté, Kendall a proposé en 1953 une notation devenue universelle [4]. Un système s'écrit $A/S/c$, où A désigne la loi des temps entre arrivées, S la loi des durées de service et c le nombre de serveurs. Les lettres usuelles sont M (markovien : arrivées de Poisson ou durées exponentielles, c'est-à-dire sans mémoire), D (durées déterministes) et G (loi générale, sans hypothèse particulière). Une forme étendue précise au besoin la capacité de la file et la discipline de service, mais le triplet suffit ici.

Le modèle M/M/1, qui combine arrivées de Poisson, services exponentiels et serveur unique, est le plus simple de tous et se résout entièrement par le calcul [1]. Lorsque $\lambda < \mu$, le nombre moyen de clients présents dans le système vaut

$$L = \frac{\rho}{1 - \rho}, \quad \rho = \frac{\lambda}{\mu},$$

et le temps moyen passé dans le système s'écrit $W = 1/(\mu - \lambda)$. Malgré son dépouillement, ce modèle capture déjà l'essentiel du comportement qualitatif d'une file : stabilité conditionnée par $\rho < 1$, croissance non linéaire de l'attente avec la charge. Son extension naturelle est le modèle M/M/c, où c serveurs identiques puisent dans une file commune ; l'analyse, plus lourde, passe par la formule d'Erlang C mais obéit à la même logique [2]. Un service d'urgences considéré globalement, avec une salle d'attente et plusieurs boxes, s'approche en première analyse d'un M/M/c. L'intérêt de ces modèles n'est pas la prédiction exacte, que leurs hypothèses interdisent, mais les ordres de grandeur et les lois qualitatives qu'ils révèlent.

1.2.3 Taux d'occupation du système

La grandeur la plus parlante issue de ces modèles est le taux d'occupation

$$\rho = \frac{\lambda}{\mu}$$

pour un serveur unique, et $\rho = \lambda/(c\mu)$ pour c serveurs. Il s'interprète comme la fraction du temps pendant laquelle les serveurs sont occupés, ou encore comme le rapport entre la charge entrante et la capacité de traitement [2].

Trois régimes se distinguent. Tant que $\rho < 1$, le système est stable : la file se vide régulièrement

et l'attente reste bornée en moyenne. Au voisinage de $\rho \approx 1$, le système entre dans un régime critique. Il absorbe encore le flux, mais le moindre aléa, qu'il s'agisse d'une arrivée groupée ou d'un service inhabituellement long, crée une file qui met très longtemps à se résorber, et l'attente moyenne s'emballe : dans le M/M/1, $W = 1/(\mu - \lambda)$ diverge lorsque λ tend vers μ . Les délais ne croissent pas proportionnellement à la charge, ils explosent. Un exemple chiffré le montre mieux qu'un théorème : pour une capacité de dix patients par heure, un flux de huit ($\rho = 0,8$) donne un temps de passage moyen de trente minutes ; un flux de neuf et demi ($\rho = 0,95$) le porte à deux heures. Lorsque enfin $\rho > 1$, le système est instable : les arrivées dépassent durablement la capacité, la file croît sans limite, et seuls l'abandon des clients ou la fermeture du service l'arrêtent [1].

La lecture hospitalière est immédiate. Un service d'urgences fonctionne souvent, aux heures de pointe, au voisinage de $\rho = 1$: c'est précisément le régime où quelques patients de plus suffisent à faire passer l'attente de trente minutes à plusieurs heures. Le personnel constate ce basculement chaque soir d'affluence ; la théorie montre qu'il ne suppose aucune défaillance particulière, puisqu'il relève de la mécanique même de la saturation [2]. Suivre le taux d'occupation donne ainsi aux responsables un signal d'alerte avant que les délais ne deviennent visibles.

1.2.4 Loi de Little

Parmi les résultats généraux de la théorie, la loi établie par Little en 1961 occupe une place à part [5]. Elle relie trois grandeurs moyennes d'un système en régime stationnaire :

$$L = \lambda W,$$

où L est le nombre moyen de clients présents, λ le débit d'arrivée et W le temps moyen passé dans le système. La même relation vaut pour la file d'attente seule : $L_q = \lambda W_q$, avec L_q le nombre moyen de clients en attente et W_q leur attente moyenne. Ce qui rend ce résultat remarquable est sa généralité : il ne suppose rien sur les lois d'arrivée ou de service, rien sur la discipline, rien sur le nombre de serveurs [5].

Son intérêt pratique aux urgences tient à une asymétrie de mesure. Deux des trois grandeurs s'observent facilement : on compte les patients présents et on lit le registre des admissions, puis la troisième s'en déduit. Un service qui compte en moyenne vingt-quatre patients présents et en admet huit par heure les retient donc trois heures en moyenne, sans avoir chronométré personne. Inversement, un objectif de temps de passage se traduit immédiatement en nombre maximal de patients à maintenir dans le service. Le tableau de bord de BetterEMS s'appuie sur cette relation : il expose en continu le nombre de patients en file et le rythme des admissions, ce qui permet d'estimer les délais courants d'un simple rapport.

1.2.5 Limites des modèles classiques

Les hypothèses qui rendent ces modèles calculables sont aussi ce qui les éloigne du terrain. La stationnarité d'abord : un λ constant ne décrit ni un flux qui double entre quatorze heures et vingt heures, ni les pics saisonniers des épidémies. L'indépendance des arrivées ensuite : un accident de la route livre plusieurs blessés d'un coup, en contradiction directe avec le modèle de Poisson. La forme exponentielle des services enfin, que la dispersion réelle des durées de soins respecte mal [3].

Aucune de ces approximations n'est pourtant la limite décisive. La vraie faiblesse des modèles classiques aux urgences tient à une hypothèse si naturelle qu'elle passe inaperçue : celle de clients interchangeables. Toute la théorie présentée jusqu'ici optimise l'attente moyenne, tous clients confondus ; or aux urgences le coût d'une minute d'attente varie de plusieurs ordres de grandeur d'un patient à l'autre. Une heure de plus en salle d'attente est sans conséquence pour une entorse ; elle peut être fatale dans un sepsis. L'attente moyenne est donc un indicateur presque vide de sens clinique : ce qui compte, c'est l'attente des plus graves. Réintroduire cette information dans le modèle, c'est exactement ce que font les files à priorité.

1.3 Files d'attente prioritaires

Une file à priorité attache à chaque client une classe, disons de 1, la plus urgente, à n , la moins urgente, et sert toujours la classe la plus urgente présente dans la file ; à classe égale, l'ordre d'arrivée départage [3]. Le principe tient en une phrase. Toute la difficulté se loge dans deux questions : comment attribuer les classes, et que faire du service en cours quand un client plus prioritaire se présente.

1.3.1 La discipline FIFO et ses limites aux urgences

Avant de critiquer le FIFO, rappelons le cadre dans lequel il opère. La figure 1.1 représente la structure générale d'un système de file d'attente : un flux d'arrivées, une file régie par une discipline de service, un ou plusieurs serveurs, des départs.

Le FIFO a de réelles vertus : aucune information requise sur les clients, aucune contestation possible, une équité immédiatement lisible. C'est la règle de la boulangerie, et elle y convient parce que tous les clients y ont à peu près le même besoin et le même coût d'attente.

Transposons-la aux urgences. Trois patients se présentent à quelques minutes d'intervalle : le premier souffre d'une douleur légère au poignet après une chute, le deuxième d'une fièvre modérée. Le troisième arrive en arrêt cardio-respiratoire. Dans une file strictement chronologique, ce dernier est servi en troisième position, après l'entorse et après la fièvre. Le cas est volontairement extrême, car aucune équipe réelle ne laisserait la règle s'appliquer jusqu'au bout, mais il met à nu le défaut de structure : le FIFO ignore la seule information qui importe ici, la gravité. Entre

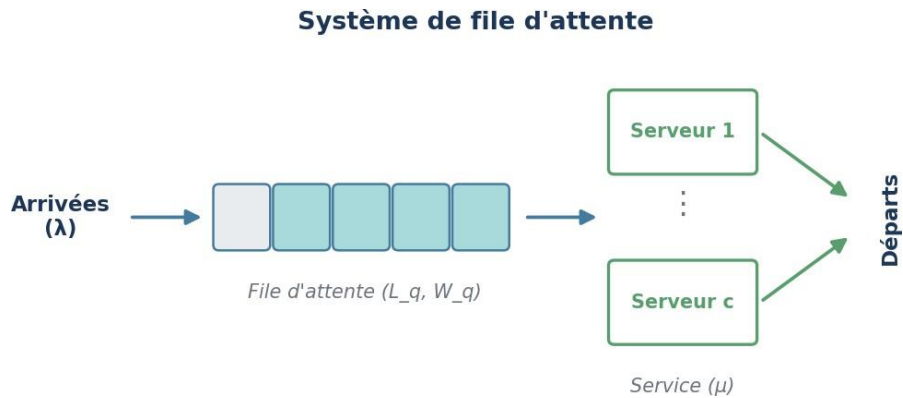


Fig. 1.1 : Structure générale d'un système de file d'attente.

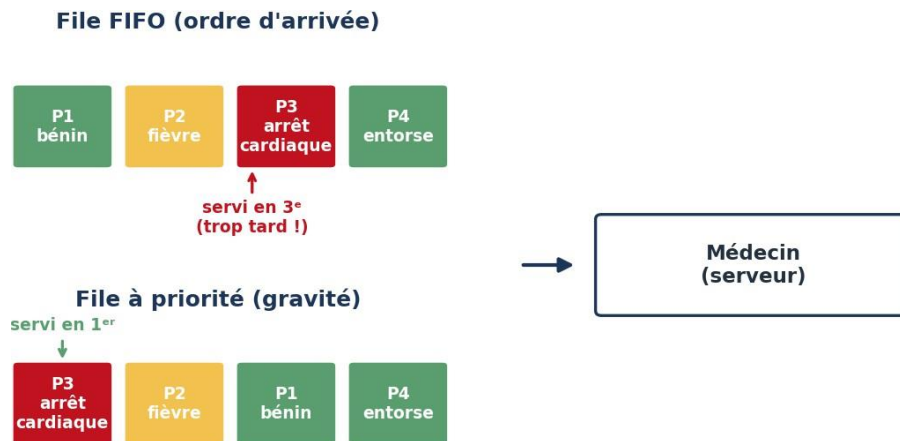


Fig. 1.2 : Discipline FIFO et file à priorité : positionnement d'un cas critique.

l'arrêt cardiaque et l'entorse, toute une gamme de situations intermédiaires, comme une douleur thoracique atypique ou une céphalée inhabituellement brutale, peut traverser une file aveugle sans alerter personne ; c'est là que le danger est réel [10].

La figure 1.2 résume la comparaison : pour une même heure d'arrivée, le cas critique occupe la troisième position dans une file FIFO et la première dans une file à priorité.

1.3.2 Priorité préemptive et non préemptive

Reste la seconde question : le sort du service en cours. La théorie distingue deux régimes [2]. Dans une file à priorité préemptive, l'arrivée d'un client plus prioritaire interrompt immédiatement le service en cours ; le client interrompu retourne en file, et son service reprend, ou recommence selon la variante, quand une ressource se libère. Dans une file non préemptive, le service entamé va à son terme quoi qu'il arrive ; la priorité ne joue que sur l'ordre des clients en attente.

Chaque régime a son terrain. La préemption minimise l'attente de la classe critique et s'impose

partout où chaque seconde compte : les ordonnanceurs de systèmes d'exploitation préemptent sans état d'âme. Elle a un coût, celui des interruptions : travail fragmenté, reprise approximative de ce qui était en cours. En milieu hospitalier, ce coût devient clinique : interrompre un geste en cours peut mettre en danger le patient qui est sur la table. La discipline non préemptive offre la garantie inverse, continuité des soins et stabilité de l'organisation, au prix d'un délai supplémentaire pour le cas critique qui survient au mauvais moment [3].

Les services d'urgences réels n'appliquent ni l'un ni l'autre à l'état pur. Leur fonctionnement quotidien est non préemptif, puisqu'un médecin termine sa consultation avant d'appeler le patient suivant, mais une urgence vitale déclenche une préemption de fait : l'équipe abandonne séance tenante les tâches qui peuvent l'être. Ce basculement n'est pas algorithmique; il relève d'un jugement humain sur ce qui, à cet instant, peut être interrompu. Nous avons reproduit cette logique hybride dans BetterEMS : la file y est non préemptive, un cas classé rouge prend la tête de la file et déclenche une alerte visuelle et sonore, mais le système n'interrompt jamais de lui-même une consultation en cours, car la décision d'interrompre appartient au médecin.

1.3.3 Effet des priorités sur les temps d'attente

L'effet d'une discipline à priorité se lit directement sur les temps d'attente moyens par classe : si W_k désigne l'attente moyenne de la classe k , alors

$$W_1 < W_2 < W_3 < \dots$$

dans tout système à priorité sous charge [3]. Il faut cependant se garder d'y voir un gain net. Une priorité ne crée aucune capacité de traitement : la charge totale du système est inchangée, et avec elle la quantité globale d'attente à distribuer. Tout ce que gagnent les classes hautes est payé, minute pour minute, par les classes basses [2].

Poussée à l'extrême, cette redistribution produit le phénomène de famine (*starvation*) : sous charge soutenue, les clients de basse priorité peuvent attendre indéfiniment, le flux des prioritaires ne tarissant jamais assez longtemps pour qu'ils soient servis. Aux urgences, le phénomène n'a rien de théorique. Un soir d'affluence, le patient venu pour un problème bénin voit passer devant lui tous les cas plus graves et peut rester des heures en salle d'attente. La situation est à la fois injuste et dangereuse, puisqu'un état clinique peut se dégrader pendant l'attente.

Deux garde-fous classiques répondent à ce risque. Le premier est le vieillissement (*aging*) : la priorité d'un client augmente avec son temps d'attente, ce qui finit mécaniquement par le faire servir. Le second consiste à garantir à chaque classe un délai maximal de prise en charge, dont l'approche ou le dépassement déclenche une alerte et une réévaluation. C'est cette seconde solution que retiennent les systèmes de triage modernes, sous la forme de délais cibles attachés à chaque niveau; le tableau 1.1 en donne la structure typique. Nous l'appliquons telle quelle dans BetterEMS, où chaque patient reçoit, avec son niveau de priorité, un délai maximal d'attente que la file surveille.

Tab. 1.1 : Délais cibles de prise en charge par niveau de priorité (valeurs inspirées du Manchester Triage System [6]).

Niveau de priorité	Exemple de situation	Délai cible de prise en charge
1 : Critique	Arrêt cardio-respiratoire	Immédiat
2 : Très urgent	Douleur thoracique intense	≤ 10 minutes
3 : Urgent	Fracture fermée, douleur marquée	≤ 60 minutes
4 : Standard	Entorse, plaie simple	≤ 120 minutes
5 : Non urgent	Demande sans caractère aigu	≤ 240 minutes

Ces délais changent la nature de la promesse faite aux patients : «les urgences passent d'abord» devient un engagement vérifiable, classe par classe, et fournit du même coup un indicateur de qualité du service, à savoir le taux de patients pris en charge dans le délai de leur niveau.

1.4 Le triage médical aux urgences

La théorie fournit le cadre ; le triage en est l'application clinique. Trier, c'est construire la fonction qui affecte chaque patient à une classe de priorité, et cette fonction n'a rien d'un calcul : c'est une décision médicale, prise en quelques minutes, sur des informations incomplètes.

1.4.1 Définition et origines

Le mot lui-même vient du français *trier*, et la pratique est née sur les champs de bataille. Les chirurgiens militaires des campagnes napoléoniennes, Dominique Larrey en tête, ont imposé l'idée de soigner les blessés selon la gravité de leurs blessures plutôt que selon leur grade ou leur ordre d'évacuation, dans une logique de masse et de pénurie où les ressources chirurgicales ne suffisent jamais à tous. C'est cette logique qui a été transposée aux urgences civiles dans la seconde moitié du XX^e siècle, à mesure que leur fréquentation croissait plus vite que leurs moyens [10].

Dans sa forme moderne, le triage se définit comme l'évaluation clinique initiale, rapide et structurée, qui attribue à chaque patient un niveau de priorité déterminant son délai maximal de prise en charge [6]. Chaque mot de cette définition compte, mais le plus important est « initiale » : le triage n'est pas un diagnostic. La question posée n'est pas « de quoi souffre ce patient ? » mais « combien de temps peut-il attendre sans danger ? ». Un triage peut être parfaitement juste sans qu'aucun diagnostic ne soit posé ; inversement, vouloir diagnostiquer au triage ferait perdre le bénéfice de sa rapidité. Concrètement, l'évaluation est conduite le plus souvent par un infirmier d'accueil expérimenté, en quelques minutes, à partir du motif de consultation, des constantes vitales et d'un interrogatoire bref. De sa qualité dépend toute l'organisation en aval.

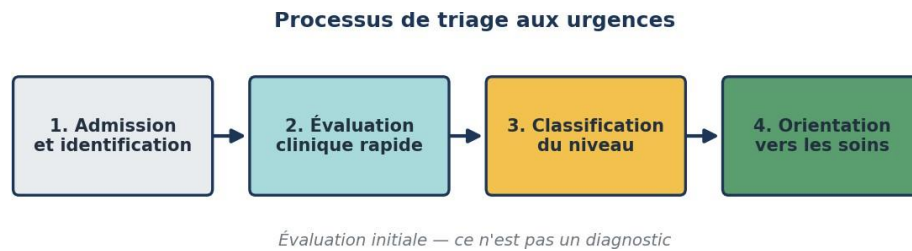


Fig. 1.3 : Processus de triage aux urgences (de l'admission à l'orientation).

1.4.2 Enjeux cliniques

Le premier enjeu est la mortalité évitable. Certaines pathologies sont dites temps-dépendantes : infarctus du myocarde, accident vasculaire cérébral, sepsis, hémorragies, dont le pronostic se dégrade avec chaque fraction d'heure de retard thérapeutique. Le triage existe d'abord pour que ces patients-là ne patientent pas. Identifier en deux minutes, dans un flux indifférencié, celui qui ne peut pas attendre : c'est la mission première, et la justification suffisante, de tout système de triage [10].

Le deuxième enjeu est l'allocation des ressources. Le box de déchocage, le plateau d'imagerie, l'attention du médecin senior sont des ressources rares ; le triage les oriente vers les patients qui en ont besoin maintenant, et protège le service d'une mobilisation désordonnée.

Un troisième enjeu, plus discret, concerne la sécurité de la salle d'attente elle-même. Dire qui peut attendre est aussi engageant que dire qui ne le peut pas : un patient classé non urgent est un patient dont on affirme, sur des critères explicites, que l'attente ne le mettra pas en danger. Cette affirmation n'est valable qu'à l'instant où elle est faite, car un état clinique évolue, ce qui fonde la pratique de la réévaluation périodique des patients en attente [6].

1.4.3 Déroulement du triage

D'un établissement à l'autre, le processus suit la même trame en quatre temps, que la figure 1.3 schématise. L'admission d'abord : identification du patient, recueil du motif de consultation, ouverture du dossier. L'évaluation ensuite : mesure des constantes vitales (fréquence cardiaque, pression artérielle, fréquence respiratoire, saturation en oxygène, température), cotation de la douleur, interrogatoire ciblé sur les symptômes et les antécédents saillants. Vient alors la classification : la confrontation de ces éléments aux critères d'un protocole produit le niveau de priorité. L'orientation enfin : déchocage pour les cas critiques, box d'examen, filière rapide pour les cas légers ou salle d'attente, avec un délai cible et des réévaluations programmées.

Dans BetterEMS, ces quatre temps structurent directement le parcours applicatif : l'agent d'accueil enregistre le patient, l'infirmier saisit constantes et symptômes, le système propose une classification que le médecin valide ou corrige, et la file d'attente ordonne l'orientation.

Nous détaillerons cette correspondance au chapitre 3.

1.4.4 Erreurs de triage : sous-triage et sur-triage

Le triage est une classification, et toute classification se trompe parfois. Deux types d'erreur sont possibles, et leurs conséquences n'ont rien de symétrique [10].

Le sous-triage consiste à sous-estimer la gravité : un patient grave reçoit un niveau de priorité inférieur à celui que son état exige. C'est l'erreur dangereuse. Elle envoie en salle d'attente un patient qui aurait dû être pris en charge sans délai, retarde le traitement d'une pathologie qui ne pardonne pas le retard, et peut coûter une vie. Les présentations atypiques en sont le terrain de prédilection : l'infarctus sans douleur thoracique franche, le sepsis débutant dont les constantes restent presque normales.

Le sur-triage est l'erreur inverse : surestimer la gravité et attribuer un niveau trop élevé. Personne n'est directement mis en danger, mais le coût organisationnel est réel : des ressources lourdes mobilisées pour un cas qui n'en avait pas besoin, la file des priorités hautes engorgée, l'attente allongée pour les patients réellement urgents. À dose élevée, le sur-triage finit par annuler le triage lui-même : si tout le monde est urgent, plus personne ne l'est.

De cette asymétrie découle le principe de précaution du triage : dans le doute, classer au-dessus. Un sur-triage ponctuel coûte de l'efficacité ; un sous-triage coûte parfois une vie, et tous les protocoles modernes assument ce biais prudent [6]. Encore faut-il que la règle soit appliquée uniformément, et c'est là que le triage purement humain atteint ses limites. Appliqué par des soignants fatigués, pressés, inégalement formés, le « dans le doute, classer au-dessus » dérive : trop large chez l'un, qui sur-trie massivement, trop restrictif chez l'autre, qui laisse passer des cas graves. Un système automatisé applique la même règle de précaution au centième patient de la nuit qu'au premier de la matinée. C'est à nos yeux l'argument le plus fort en faveur d'une standardisation automatisée du triage : non pas remplacer le jugement clinique, mais garantir que la règle de prudence collectivement choisie est appliquée de façon constante.

1.4.5 Limites du triage manuel

La subjectivité vient en premier. Le même patient, présenté à deux infirmiers de triage, ne reçoit pas toujours le même niveau : l'expérience de chacun, sa spécialité d'origine, sa sensibilité personnelle au risque pèsent sur la décision. Les études de fiabilité menées sur les échelles de triage mesurent cette variabilité inter-opérateurs et montrent qu'elle reste substantielle même avec des protocoles bien établis [7]. Ce n'est pas une question de compétence, mais la conséquence inévitable d'une décision rapide prise sur des informations partielles.

La fatigue cognitive vient ensuite. Le triage se pratique dans le bruit, sous l'interruption permanente, et souvent la nuit. Une décision répétée des dizaines de fois par garde voit sa qualité décliner avec la charge ; c'est vrai de toute tâche cognitive, et rien n'exempte le triage de cette

loi [10]. Les périodes de forte affluence cumulent ainsi les deux risques : c'est quand le triage importe le plus que les conditions de son exercice sont les pires.

La standardisation, enfin, reste difficile même quand un protocole existe. Son application varie d'un établissement à l'autre, d'une équipe à l'autre, selon la formation reçue et les habitudes locales ; le protocole le mieux conçu ne standardise que ce que ses utilisateurs en appliquent réellement.

Aucune de ces limites ne disqualifie le triage humain, qui reste le socle de l'organisation des urgences. Elles dessinent en revanche, en creux, le cahier des charges d'une assistance : un outil qui propose une classification constante, insensible à l'heure et à la fatigue, appliquant strictement le même protocole à chaque patient, et que le soignant reste libre de corriger. C'est la définition même d'un système d'aide à la décision, et les progrès de l'intelligence artificielle, que le chapitre 2 retracera, rendent aujourd'hui un tel outil réalisable. Encore faut-il choisir le protocole de référence à automatiser : c'est l'objet de la section suivante.

1.5 Les systèmes de triage normalisés

À partir des années 1990, plusieurs pays ont entrepris de codifier le triage dans des échelles normalisées à cinq niveaux, les travaux d'évaluation ayant établi que cinq niveaux offrent une fiabilité et une validité supérieures aux échelles à trois niveaux utilisées jusque-là [7]. Trois systèmes dominant aujourd'hui la pratique internationale : le Manchester Triage System britannique, l'Emergency Severity Index américain et la Canadian Triage and Acuity Scale [10]. Ils poursuivent le même but sans emprunter le même chemin, une différence qui pèse lourd dès lors qu'on envisage une automatisation.

1.5.1 Manchester Triage System

Développé au Royaume-Uni au milieu des années 1990 par le Manchester Triage Group, le MTS est devenu le système de référence dans une grande partie de l'Europe [6]. Son mécanisme repose sur une cinquantaine d'arbres décisionnels (*flowcharts*), un par grand motif de consultation : douleur thoracique, céphalée, traumatisme de membre, douleur abdominale, et ainsi de suite. L'infirmier sélectionne l'arbre correspondant au motif du patient, puis parcourt une liste ordonnée de discriminateurs, c'est-à-dire des critères cliniques explicites, soit généraux (risque vital immédiat, détresse respiratoire, hémorragie incontrôlable, altération de la conscience, douleur intense, température), soit propres au motif. Le premier discriminateur vérifié détermine le niveau de priorité [6]. La liste étant parcourue du plus grave au moins grave, le mécanisme intègre par construction le principe de précaution évoqué plus haut.

Le résultat est une classification en cinq niveaux, chacun associé à une couleur et à un délai cible de prise en charge, que le tableau 1.2 détaille et que la figure 1.4 illustre.

Les cinq niveaux du Manchester Triage System

	Niveau 1 — Rouge	Immédiat (0 min)
	Niveau 2 — Orange	Très urgent (≤ 10 min)
	Niveau 3 — Jaune	Urgent (≤ 60 min)
	Niveau 4 — Vert	Standard (≤ 120 min)
	Niveau 5 — Bleu	Non urgent (≤ 240 min)

Fig. 1.4 : Les cinq niveaux du Manchester Triage System et leurs délais cibles.

Tab. 1.2 : Les cinq niveaux du Manchester Triage System [6].

Niveau	Couleur	Désignation	Délai cible
1	Rouge	Immédiat	0 minute
2	Orange	Très urgent	≤ 10 minutes
3	Jaune	Urgent	≤ 60 minutes
4	Vert	Standard	≤ 120 minutes
5	Bleu	Non urgent	≤ 240 minutes

Les forces du MTS tiennent à sa formalisation. Des critères explicites, parcourus dans un ordre fixe, laissent peu de place à l'interprétation : c'est le plus reproductible des trois systèmes, et celui dont l'application se vérifie le plus facilement a posteriori. Ses délais cibles, exprimés en minutes, se branchent directement sur la gestion de la file d'attente, chaque couleur étant à la fois un niveau clinique et une consigne opérationnelle. Sa large adoption européenne a par ailleurs produit une littérature d'évaluation abondante. Ses limites sont celles de tout protocole : la classification ne vaut que ce que vaut le recueil initial, car un discriminateur non recherché est un discriminateur non vérifié, les présentations atypiques peuvent se glisser entre les mailles des arbres, et l'évaluation de la douleur conserve une part irréductible de subjectivité [6].

1.5.2 Emergency Severity Index

L'Emergency Severity Index, conçu aux États-Unis à la fin des années 1990 [7] et maintenu depuis sous l'égide de l'AHRQ [9], suit une logique sensiblement différente. Pas d'arbres par motif ni de délais cibles : un algorithme unique, parcouru en quelques questions. Le patient nécessite-t-il une intervention vitale immédiate ? Il est classé ESI 1. Présente-t-il une situation

à haut risque, une confusion, une détresse ou une douleur sévère? ESI 2. Sinon, le classement bascule sur un critère original : combien de ressources distinctes (examens biologiques, imagerie, gestes techniques, avis spécialisés) sa prise en charge va-t-elle vraisemblablement consommer ? Plusieurs ressources mènent à ESI 3, sous réserve de constantes vitales acceptables ; une seule à ESI 4 ; aucune à ESI 5 [9].

Tab. 1.3 : Les cinq niveaux de l'Emergency Severity Index [9].

Niveau	Désignation	Critère principal
ESI 1	Réanimation	Intervention vitale immédiate requise
ESI 2	Très urgent	Situation à haut risque, détresse, douleur sévère
ESI 3	Urgent	Plusieurs ressources prévisionnelles
ESI 4	Semi-urgent	Une seule ressource prévisionnelle
ESI 5	Non urgent	Aucune ressource prévisionnelle

Cette prévision des ressources est la signature de l'ESI : le système trie autant par charge prévisible que par gravité, ce qui facilite l'orientation des cas légers vers des filières rapides et donne une vision anticipée de la charge du service. Elle en est aussi le talon d'Achille. Estimer les ressources qu'exigera un patient suppose une expérience clinique solide et dépend des pratiques locales de prescription : deux évaluateurs peuvent légitimement diverger sur le même cas [9]. L'absence de délais cibles prive en outre la file d'attente d'un critère de surveillance directement exploitable.

1.5.3 Canadian Triage and Acuity Scale

La Canadian Triage and Acuity Scale, introduite à la fin des années 1990 pour uniformiser les pratiques des urgences canadiennes [8], revient à la logique des délais cibles, avec des valeurs plus resserrées que celles du MTS. L'évaluation part d'un motif de consultation choisi dans une liste normalisée, puis applique des modificateurs (constantes vitales, intensité de la douleur, mécanisme lésionnel, terrain du patient) qui ajustent le niveau à la hausse ou à la baisse. Le système est adopté à l'échelle nationale, révisé périodiquement et décliné en version pédiatrique, ce qui en fait l'un des cadres de triage les plus aboutis [8].

Tab. 1.4 : Les cinq niveaux de la Canadian Triage and Acuity Scale [8].

Niveau	Désignation	Délai cible
CTAS I	Réanimation	Immédiat
CTAS II	Très urgent	≤ 15 minutes
CTAS III	Urgent	≤ 30 minutes
CTAS IV	Semi-urgent	≤ 60 minutes
CTAS V	Non urgent	≤ 120 minutes

Cette richesse a un prix. La CTAS exige une formation approfondie, son évaluation est parfois plus longue que celle de ses concurrents, et son fonctionnement, fondé sur des listes normalisées, des modificateurs et un audit des délais, est étroitement lié à l'organisation et à l'informatisation du système de santé canadien, ce qui complique sa transposition dans des environnements aux moyens plus contraints [8].

1.5.4 Comparaison et choix pour BetterEMS

Les trois systèmes convergent sur l'essentiel : cinq niveaux, primauté du risque vital, biais assumé vers la prudence. Ils divergent en revanche sur la logique décisionnelle, comme le résume le tableau 1.5.

Tab. 1.5 : Comparaison des trois systèmes de triage normalisés [6], [8], [9].

Critère	MTS	ESI	CTAS
Origine	Royaume-Uni	États-Unis	Canada
Nombre de niveaux	5	5	5
Logique décisionnelle	Arbres par motif + discriminateurs	Algorithme unique + ressources prévisionnelles	Motif normalisé + modificateurs
Délais cibles explicites	Oui	Non	Oui
Estimation des ressources	Non	Oui	Partielle
Formalisation des critères	Très élevée	Moyenne	Élevée
Adéquation à l'automatisation	Très élevée	Moyenne	Élevée

Pour BetterEMS, nous avons retenu le Manchester Triage System. Sa structure décisionnelle est la plus formalisée des trois : des critères explicites, ordonnés, en nombre fini, exactement

le type de connaissance qui se prête à une automatisation, qu'elle procède par règles ou par apprentissage. Ses cinq classes sont nettes, mutuellement exclusives, dotées chacune d'une couleur et d'un délai cible : elles fournissent sans aucune transformation les étiquettes d'un problème de classification à cinq classes et les clés de tri d'une file d'attente à priorité. L'ESI, à l'inverse, fait reposer ses niveaux intermédiaires sur une prévision de ressources difficile à inférer des seules données d'admission ; la CTAS, précise mais lourde, reste étroitement liée à l'organisation hospitalière canadienne. Le MTS, enfin, est éprouvé dans des environnements saturés, ce qui correspond au terrain que nous visons.

Ce choix structure tout notre travail. Les 12 400 cas cliniques de notre corpus sont annotés sur l'échelle MTS ; les cinq couleurs en sont les classes de sortie de notre modèle ; et le délai cible du protocole accompagne chaque classification jusque dans la file d'attente de l'application. Le tableau 1.2 n'est donc pas une simple référence bibliographique : c'est la spécification fonctionnelle du cœur de BetterEMS.

1.6 Les urgences hospitalières en Algérie

Le système de santé algérien repose sur un réseau d'établissements publics (centres hospitalo-universitaires, établissements publics hospitaliers, polycliniques) complété par un secteur privé en croissance. Les services d'urgences y occupent une position singulière : faute d'une médecine de premier recours toujours accessible, en particulier la nuit et les jours fériés, ils servent de porte d'entrée par défaut dans le système de soins. Une part notable des passages relève ainsi de consultations sans caractère urgent, qui s'ajoutent au flux des urgences vraies et entretiennent une surcharge chronique, particulièrement sensible dans les grands centres urbains où se concentrent les plateaux techniques.

Face à cette saturation, l'organisation de l'accueil pèse d'autant plus lourd. Or le triage reste, dans la plupart des établissements, manuel et peu formalisé : l'orientation des patients repose sur l'appréciation du personnel d'accueil, son expérience et les habitudes du service, plus que sur un protocole normalisé. Aucun des grands systèmes présentés à la section précédente ne fait l'objet, à notre connaissance, d'un déploiement standardisé et informatisé à l'échelle nationale. Les pratiques varient donc fortement d'un établissement à l'autre : certains services universitaires disposent d'une organisation d'accueil structurée, tandis que d'autres structures fonctionnent avec des moyens très contraints, et cette hétérogénéité rend la qualité du triage dépendante du lieu et de l'équipe de garde.

La numérisation des processus de soins, elle aussi, reste limitée et inégale. Les systèmes d'information existants couvrent surtout la gestion administrative ; le dossier papier demeure le support dominant du parcours de soins, et les données des urgences sont rarement structurées. Les conséquences se cumulent : files de fait aveugles à la gravité aux heures d'affluence, temps d'attente longs et invérifiables faute de traçabilité, personnel qui trie sans outil, responsables privés des indicateurs les plus élémentaires pour piloter leur service.

Ce tableau, sévère, dessine aussi une occasion à saisir. Parce que le terrain est vierge, une solution de triage informatisé n'y rencontre ni système installé à remplacer, ni habitudes logicielles à désapprendre. Elle peut être conçue d'emblée pour ce contexte : en français, langue de travail du personnel soignant algérien ; adossée à un protocole international éprouvé ; déployable sur l'infrastructure propre de l'établissement, sans dépendance à un cloud étranger, un point décisif s'agissant de données de santé ; et dimensionnée pour des moyens matériels modestes. C'est ce créneau que vise BetterEMS, conçu localement pour des urgences algériennes ; l'étude entrepreneuriale du chapitre 5 reviendra sur ce positionnement.

1.7 Conclusion

Ce chapitre a établi le diagnostic qui motive tout le reste du travail. La théorie des files d'attente fournit le vocabulaire, à savoir le taux d'arrivée, le taux de service, l'occupation ρ et la loi de Little, ainsi qu'une leçon centrale : au voisinage de la saturation, régime ordinaire des urgences, les délais ne croissent pas avec la charge, ils explosent. Elle montre surtout que la discipline FIFO, équitable en apparence, est aveugle à la seule information qui compte aux urgences : la gravité. Les files à priorité corrigent ce défaut, au prix d'un risque de famine que les délais cibles par niveau permettent de contenir. Le triage médical est l'application clinique de ces files : une évaluation initiale, et non un diagnostic, qui classe chaque patient dès l'admission. Ses erreurs caractéristiques, sous-triage et sur-triage, comme les limites du triage manuel, à savoir la variabilité inter-opérateurs, la fatigue et la standardisation incomplète, plaident pour une assistance automatisée appliquant constamment la même règle de prudence. Parmi les trois protocoles normalisés dominants, nous avons retenu le Manchester Triage System : ses critères formalisés, ses cinq classes nettes et ses délais explicites en font le candidat naturel à une automatisation, et il fournit à BetterEMS ses étiquettes de classification comme les clés de tri de sa file d'attente. Le contexte algérien, enfin, conjugue un besoin réel et un terrain favorable à une solution conçue localement.

Reste la question que ce chapitre laisse ouverte : comment automatiser la classification elle-même ? Affecter un patient à l'un des cinq niveaux du MTS à partir d'un motif de consultation, de constantes vitales et de quelques phrases de description clinique est un problème de classification, mais sur des données en grande partie textuelles, en français, dans un domaine où l'erreur coûte cher. Le chapitre 2 passe en revue les outils disponibles pour le résoudre, de l'apprentissage automatique classique aux grands modèles de langue médicaux, et aboutit au choix technique qui fonde notre système : l'adaptation d'un modèle de langue médical spécialisé à la tâche du triage.

État de l'art : de l'apprentissage automatique aux modèles de langage médicaux

2.1 Introduction

Le chapitre précédent a ramené le triage des urgences à une question précise : attribuer à chaque patient, dès son admission, l'un des cinq niveaux du protocole de Manchester. Posé en ces termes, le problème devient un problème de classification, exactement le type de tâche que l'apprentissage automatique sait résoudre, à condition de disposer d'exemples en nombre suffisant et d'une manière fiable de mesurer les erreurs.

Ce chapitre construit le socle technique de notre travail en suivant une progression simple. Nous partons des notions de base de l'apprentissage automatique : ce qu'« apprendre » signifie pour une machine, comment on mesure ce qui a été appris, et ce qui peut mal tourner en cours d'apprentissage. Nous descendons ensuite dans les réseaux de neurones, dont les grands modèles de langage ne sont qu'un prolongement à très grande échelle. Le traitement automatique du langage naturel occupe la section suivante, car la plainte d'un patient s'exprime d'abord en texte libre ; il mène à l'architecture Transformer, puis aux grands modèles de langage et à leurs déclinaisons médicales. Les techniques d'adaptation efficace de ces modèles (LoRA, QLoRA, quantification) précèdent une revue des travaux ayant appliqué l'IA au triage des urgences, revue qui nous permettra, en fin de chapitre, de situer BetterEMS par rapport à l'existant.

2.2 Apprentissage automatique

2.2.1 Définition et paradigmes

La définition la plus citée de l'apprentissage automatique est due à Tom Mitchell : un programme apprend de l'expérience E , relativement à une classe de tâches T et à une mesure de performance P , si sa performance sur les tâches de T , mesurée par P , s'améliore avec l'expérience E [13]. Derrière cette formulation un peu sèche se cache un renversement complet de la programmation classique. Un programme traditionnel applique des règles écrites à la main pour transformer des entrées en sorties ; un système d'apprentissage reçoit des entrées accompagnées des sorties attendues et en déduit lui-même les règles, sous la forme d'une fonction mathématique ajustée aux données [11].

On distingue habituellement trois paradigmes selon la nature de l'expérience disponible. En apprentissage supervisé, chaque exemple d'entraînement est annoté : le système voit à la fois la situation et la bonne réponse, et il apprend à reproduire cette correspondance sur des cas nouveaux. C'est le paradigme dominant des applications concrètes, qu'il s'agisse de classification (prédire une catégorie) ou de régression (prédire une valeur numérique) [11]. L'apprentissage non supervisé travaille au contraire sur des données brutes, sans annotation : le système y cherche des structures cachées, par exemple des groupes de patients aux profils cliniques voisins. Enfin, l'apprentissage par renforcement ne repose ni sur des annotations ni sur des regroupements, mais sur l'interaction : un agent agit dans un environnement, reçoit des récompenses ou des pénalités, et ajuste progressivement sa stratégie [14]. Ce dernier paradigme reste marginal dans les systèmes cliniques opérationnels, où l'on ne peut évidemment pas apprendre par essais-erreurs sur des patients réels.

Le triage de Manchester relève sans ambiguïté du premier paradigme. La tâche T consiste à attribuer un niveau parmi cinq (Rouge, Orange, Jaune, Vert, Bleu) à un dossier d'admission ; l'expérience E est un ensemble de cas de triage déjà résolus, chacun portant le niveau retenu par un soignant ; la performance P se mesure par la concordance entre les niveaux prédits et les niveaux de référence [6]. Il s'agit donc d'une classification supervisée à cinq classes, et tout notre travail, y compris lorsqu'il mobilise un modèle de langage de plusieurs milliards de paramètres, reste fidèle à ce cadre.

2.2.2 Concepts fondamentaux

Un exemple d'apprentissage supervisé associe des variables d'entrée, les *features*, à une variable de sortie, le *label*. Dans notre projet, un cas de triage typique se présente ainsi : un homme de 62 ans, fréquence cardiaque à 118 battements par minute, saturation en oxygène à 89 %, douleur thoracique irradiant vers le bras gauche cotée 8/10, antécédent d'hypertension. Les features sont l'âge, le sexe, les constantes vitales, le niveau de douleur et la description des symptômes ; le label est le niveau de Manchester attendu, ici Rouge (la saturation à 89 % satisfait à elle seule le discriminant de détresse respiratoire $SpO_2 < 90 \%$, en plus de la douleur thoracique d'allure cardiaque). Le modèle n'apprend rien d'autre que la relation statistique entre ces deux blocs : des milliers de paires (features, label) lui suffisent à approcher le raisonnement qui mène des constantes et de la plainte au niveau de gravité.

Encore faut-il évaluer honnêtement ce qui a été appris. La pratique standard découpe les données en trois ensembles disjoints [12]. L'ensemble d'entraînement sert à ajuster les paramètres du modèle. L'ensemble de validation, jamais utilisé pour l'ajustement, sert à surveiller l'apprentissage en cours de route et à choisir les hyperparamètres ou le meilleur point d'arrêt. L'ensemble de test, mis sous clé jusqu'à la toute fin, fournit l'estimation finale et impartiale de la performance. Nous avons appliqué ce protocole à la lettre : nos 12 600 cas de triage rédigés manuellement ont été répartis en 12 400 cas d'entraînement, 100 de validation et 100 de test, avec une graine

aléatoire fixée (seed 42) pour garantir la reproductibilité du découpage.

Ce protocole n'a qu'un seul but : estimer la capacité de généralisation du modèle, c'est-à-dire sa performance sur des cas qu'il n'a jamais vus. Un modèle qui réciterait parfaitement ses 12 400 cas d'entraînement mais se tromperait sur le premier cas inédit serait inutilisable aux urgences, où chaque patient est, par définition, un cas nouveau. La généralisation est la seule propriété qui compte en production ; tout le reste (architecture, optimisation, volume de données) n'est que moyen d'y parvenir [11].

2.2.3 Surapprentissage et sous-apprentissage

Deux pathologies symétriques guettent tout apprentissage. Le sous-apprentissage (*underfitting*) survient quand le modèle est trop simple pour la tâche : il n'arrive même pas à bien classer les exemples d'entraînement, et ses erreurs sur les données nouvelles sont du même ordre. Un modèle linéaire qui tenterait de prédire le niveau de triage à partir de la seule fréquence cardiaque serait dans ce cas : la gravité clinique dépend d'interactions bien plus riches entre constantes, symptômes et terrain du patient.

Le surapprentissage (*overfitting*) est le défaut inverse, et le plus insidieux. Un modèle doté d'une grande capacité finit par mémoriser les particularités de l'ensemble d'entraînement (son bruit, ses formulations récurrentes, ses cas atypiques) au lieu d'en extraire les régularités générales [12]. Les signes sont caractéristiques : la perte d'entraînement continue de baisser, tandis que la perte de validation, après être descendue, se met à remonter. Le modèle devient meilleur sur ce qu'il connaît déjà et pire sur tout le reste.

La taille du jeu de données pèse directement sur ce risque. Plus le rapport entre la capacité du modèle et le nombre d'exemples est déséquilibré, plus la mémorisation devient facile et tentante : un réseau de plusieurs milliards de paramètres ajusté sur quelques milliers de cas se trouve exactement dans cette configuration délicate. Les parades classiques sont connues : régularisation, *dropout* (désactivation aléatoire de neurones pendant l'entraînement), augmentation artificielle des données, et surtout arrêt au bon moment : on surveille la perte de validation à chaque époque et l'on conserve le point de contrôle où elle est minimale, plutôt que le dernier [12]. Nous aurons l'occasion de constater, sur nos propres courbes d'entraînement, à quel point cette discipline est tout sauf théorique.

2.2.4 Métriques d'évaluation

Pour un problème de classification, la matrice de confusion est le point de départ de toute évaluation sérieuse. Dans le cas binaire, elle croise la réalité et la prédiction en quatre cases.

Tab. 2.1 : Matrice de confusion d'un classifieur binaire.

	Prédit positif	Prédit négatif
Réellement positif	Vrai positif (TP)	Faux négatif (FN)
Réellement négatif	Faux positif (FP)	Vrai négatif (TN)

De ces quatre quantités découlent les métriques usuelles [11]. L'exactitude (*accuracy*) mesure la proportion globale de prédictions correctes :

$$\text{Exactitude} = \frac{TP + TN}{TP + TN + FP + FN}$$

La précision mesure la fiabilité des alertes positives : parmi les cas que le modèle déclare positifs, combien le sont réellement :

$$\text{Précision} = \frac{TP}{TP + FP}$$

Le rappel (ou sensibilité) mesure la couverture des cas réellement positifs, c'est-à-dire, parmi eux, combien le modèle a su détecter :

$$\text{Rappel} = \frac{TP}{TP + FN}$$

Le score F1 combine les deux par leur moyenne harmonique, ce qui pénalise les modèles qui sacrifient entièrement l'une au profit de l'autre :

$$F_1 = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

En triage, ces métriques ne se valent pas. L'exactitude globale peut être trompeuse dès que les classes sont déséquilibrées, et elles le sont toujours aux urgences : dans notre propre jeu de données manuel, la classe Bleu ne représente que 7,2 % des cas. Un modèle qui ignorerait complètement cette classe perdrait à peine quelques points d'exactitude tout en étant cliniquement défaillant. La métrique qui compte vraiment est le rappel des classes graves. Un faux négatif sur la classe Rouge, c'est très exactement le sous-triage décrit au chapitre 1 : un patient en détresse vitale classé comme pouvant attendre, avec un délai de prise en charge qui peut lui être fatal. Un faux positif (un cas bénin classé trop haut) gaspille des ressources et allonge l'attente des autres, mais ne tue personne. Cette asymétrie des coûts d'erreur doit guider à la fois l'évaluation et l'entraînement de tout système de triage automatisé : entre un modèle légèrement plus exact et un modèle au rappel plus élevé sur Rouge et Orange, c'est le second qu'il faut choisir.

2.3 Apprentissage profond

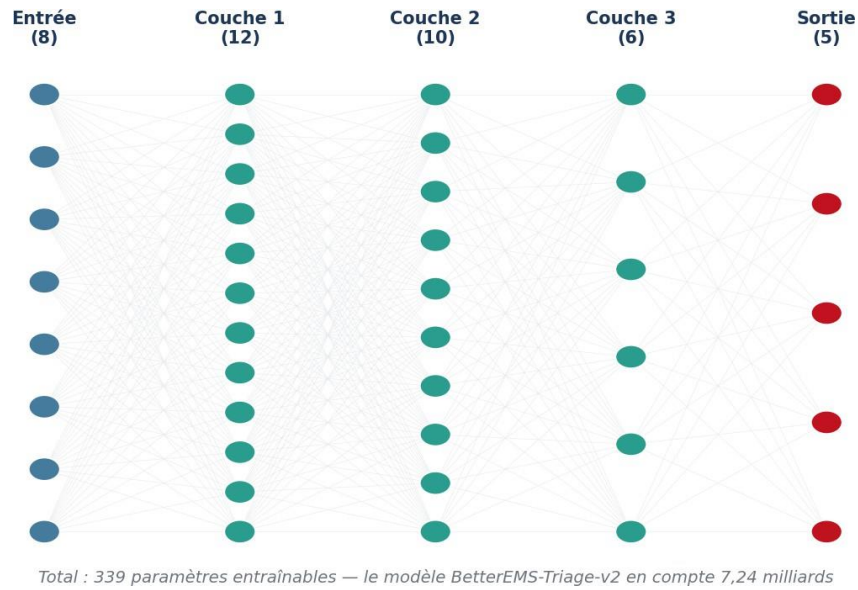


Fig. 2.1 : Réseau de neurones multicouches (8-12-10-6-5) pour une classification à cinq classes.

2.3.1 Le neurone artificiel et le perceptron

L'apprentissage profond repose sur un composant élémentaire d'une grande simplicité : le neurone artificiel. Il reçoit des entrées numériques $x_1, \dots, x_n$, les pondère par des coefficients $w_1, \dots, w_n$ appris pendant l'entraînement, ajoute un biais b , et calcule la somme :

$$z = \sum_i w_i x_i + b$$

Cette somme passe ensuite dans une fonction d'activation non linéaire f , et la sortie du neurone vaut $a = f(z)$. Le perceptron de Frank Rosenblatt (1958) est le plus ancien modèle de ce type : un neurone unique dont l'activation est un simple seuil, capable d'apprendre à séparer deux classes par un hyperplan [14]. Son pouvoir expressif est limité : un perceptron seul ne peut représenter que des frontières de décision linéaires, ce qui exclut des fonctions aussi simples que le OU exclusif. La parade tient en un mot : empiler. En composant plusieurs couches de neurones, chacune transformant la sortie de la précédente, on obtient des fonctions arbitrairement complexes ; c'est tout le principe des réseaux multicouches [12].

2.3.2 Réseaux multicouches

Un réseau multicouche (*multi-layer perceptron*, MLP) enchaîne une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie, chaque neurone d'une couche étant connecté à tous les neurones de la suivante. Le qualificatif « profond » de l'apprentissage profond vient simplement du nombre de couches cachées empilées [12].

Prenons un exemple dimensionné pour notre problème : un réseau qui recevrait huit caractéristiques d'un patient (âge, fréquence cardiaque, fréquence respiratoire, tension, saturation, température,

niveau de douleur, score de conscience) et produirait une probabilité pour chacun des cinq niveaux de Manchester. La figure 2.1 montre une architecture 8-12-10-6-5 : trois couches cachées de 12, 10 et 6 neurones entre l'entrée et la sortie. Chaque connexion porte un poids et chaque neurone un biais, ce qui donne le décompte du tableau 2.2.

Tab. 2.2 : Décompte des paramètres entraînables du réseau 8-12-10-6-5.

Transition	Calcul	Paramètres
Entrée (8) → couche cachée 1 (12)	$8 \times 12 + 12$	108
Couche cachée 1 (12) → couche cachée 2 (10)	$12 \times 10 + 10$	130
Couche cachée 2 (10) → couche cachée 3 (6)	$10 \times 6 + 6$	66
Couche cachée 3 (6) → sortie (5)	$6 \times 5 + 5$	35
Total		339

Le calcul effectué par le réseau, appelé propagation avant, s'écrit de façon uniforme pour chaque couche l :

$$h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)})$$

où $h^{(0)} = x$ est le vecteur d'entrée, $W^{(l)}$ la matrice des poids de la couche l et f la fonction d'activation. La couche de sortie remplace f par la fonction softmax, qui transforme cinq scores réels en cinq probabilités positives de somme 1 :

$$\text{softmax}(z_j) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Le réseau prédit alors la classe de probabilité maximale, par exemple Orange avec 0,71, Jaune avec 0,18, et le reste réparti sur les trois autres niveaux. Ces 339 paramètres donnent l'échelle du jouet pédagogique; le modèle de langage que nous utiliserons plus loin en compte 7,24 milliards, soit plus de vingt millions de fois plus. La mécanique, elle, reste identique : des multiplications matricielles, des additions de biais et des non-linéarités, empilées.

2.3.3 Fonctions d'activation

Sans fonction d'activation, l'empilement de couches ne servirait à rien : une composition de fonctions linéaires reste linéaire, et le réseau le plus profond s'effondrerait en un simple perceptron. La non-linéarité insérée entre les couches est donc ce qui donne au réseau son pouvoir expressif [12].

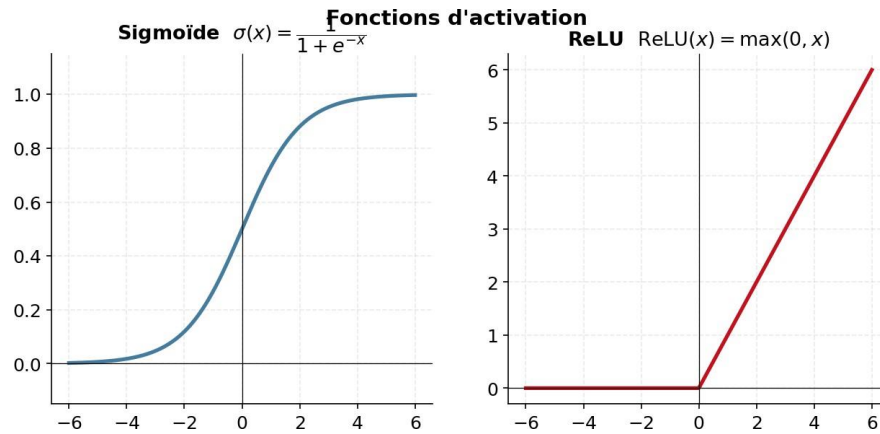


Fig. 2.2 : Fonctions d'activation sigmoïde et ReLU.

Deux fonctions dominent l'histoire du domaine. La sigmoïde écrase tout réel dans l'intervalle $]0, 1[$:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Longtemps standard, elle souffre d'un défaut sérieux pour les réseaux profonds : sa dérivée s'annule presque dès que $|x|$ dépasse quelques unités, ce qui étouffe les gradients lors de l'entraînement des couches profondes. C'est le phénomène dit du gradient évanescent. La fonction ReLU (*rectified linear unit*) la supplante dans la plupart des architectures modernes :

$$\text{ReLU}(x) = \max(0, x)$$

Nulle pour les entrées négatives, identité pour les positives, elle se calcule en une comparaison et laisse passer les gradients sans les atténuer sur tout le demi-axe positif. La figure 2.2 superpose les deux courbes. Les Transformers récents emploient des variantes lissées de ReLU (GELU, SwiGLU), mais le principe reste celui-là : une non-linéarité simple, peu coûteuse, favorable à la circulation des gradients.

2.3.4 Rétropropagation et descente de gradient

Reste à expliquer comment les poids du réseau acquièrent leurs valeurs. On définit d'abord une fonction de perte L qui quantifie l'écart entre les prédictions du réseau et les réponses attendues ; pour la classification, l'entropie croisée est le choix standard. L'entraînement consiste alors à chercher les poids qui minimisent cette perte, par descente de gradient : à chaque itération, on calcule le gradient de la perte par rapport à chaque poids, c'est-à-dire la direction dans laquelle ce poids fait le plus augmenter l'erreur, et l'on fait un petit pas dans la direction opposée :

$$w \leftarrow w - \eta \nabla_w L$$

Le calcul efficace de ces gradients dans un réseau à plusieurs couches est l'apport de l'algorithme de rétropropagation, qui applique la règle de dérivation des fonctions composées en remontant de la sortie vers l'entrée, couche par couche [12]. En pratique, le gradient n'est pas calculé sur tout le jeu de données mais sur de petits lots (*mini-batches*), et des optimiseurs adaptatifs comme Adam ou sa variante AdamW modulent le pas effectif de chaque poids selon l'historique de ses gradients.

Le coefficient η , appelé taux d'apprentissage, est l'hyperparamètre le plus sensible de tout l'entraînement. Trop grand, les mises à jour dépassent systématiquement le minimum recherché et la perte oscille, voire diverge : elle croît au lieu de décroître, et le modèle se dégrade à chaque pas. Trop petit, l'entraînement progresse à une lenteur prohibitive et peut stagner dans un creux médiocre. L'ordre de grandeur dépend fortement du régime de travail : entraîner un réseau depuis zéro tolère des taux relativement élevés, mais affiner un modèle déjà pré-entraîné exige des pas minuscules, typiquement autour de 10^{-5} , comme le taux de 2×10^{-5} que nous utiliserons, sous peine de détruire en quelques itérations des connaissances accumulées sur des milliards de mots. Nous verrons que cette sensibilité du taux d'apprentissage, combinée à d'autres facteurs, a directement pesé sur le déroulement de nos expérimentations.

2.3.5 GPU et accélération matérielle

Tout ce qui précède se résume, du point de vue du matériel, à des multiplications de matrices : la propagation avant en est faite, la rétropropagation aussi. Or un processeur classique (CPU), conçu pour exécuter rapidement des séquences d'instructions variées, n'exploite qu'une poignée de cœurs. Un processeur graphique (GPU) suit la philosophie inverse : des milliers d'unités de calcul simples qui exécutent la même opération sur des données différentes, soit exactement le profil d'un produit matriciel, où chaque coefficient de la matrice résultat se calcule indépendamment des autres. C'est cette adéquation, plus que tout progrès algorithmique, qui a rendu praticable l'entraînement de réseaux profonds à partir des années 2010 [12].

Deux ressources du GPU comptent pour notre usage. La puissance de calcul détermine la vitesse d'entraînement et d'inférence; la mémoire embarquée (VRAM) détermine ce qui peut tenir sur la carte : poids du modèle, activations intermédiaires, gradients et états de l'optimiseur. Pour les grands modèles de langage, la VRAM est presque toujours la contrainte qui mord en premier : c'est elle qui décide si un fine-tuning est possible ou non sur une machine donnée. Nos expérimentations ont été menées sur une carte NVIDIA de classe professionnelle, mise à disposition par le Centre de Recherche en Informatique de l'Université de Saïda ; sa mémoire a conditionné, comme nous le verrons, plusieurs choix techniques du projet.

2.4 Traitement automatique du langage naturel

2.4.1 Représenter le texte : du sac de mots aux embeddings

Un réseau de neurones ne manipule que des nombres ; or « douleur thoracique irradiant vers le bras gauche » n'en est pas un. Toute l'histoire du traitement automatique du langage naturel (TALN) peut se lire comme une succession de réponses à cette question : comment convertir du texte en vecteurs sans perdre le sens [15], [16] ?

La réponse la plus ancienne est le sac de mots (*bag of words*) : un document devient un vecteur de comptages, une case par mot du vocabulaire. La représentation est simple et robuste, mais elle ignore l'ordre des mots (« douleur sans fièvre » et « fièvre sans douleur » y sont identiques) et donne le même poids à « le » qu'à « hémorragie ». La pondération TF-IDF corrige ce second défaut en divisant la fréquence d'un terme dans le document par sa fréquence dans l'ensemble du corpus, ce qui met en avant les mots discriminants ; elle ne résout en rien le premier.

Le tournant suivant est celui des plongements lexicaux (*word embeddings*). L'idée de word2vec [17] est d'apprendre, pour chaque mot, un vecteur dense de quelques centaines de dimensions tel que les mots apparaissant dans des contextes similaires aient des vecteurs proches. La géométrie obtenue capture une part réelle de la sémantique : « tachycardie » se retrouve près de « palpitations », et des régularités arithmétiques émergent entre vecteurs. Ces embeddings restent pourtant statiques : un mot, un vecteur, quel que soit le contexte. La polysémie les met en échec : en français médical, « tension » désigne aussi bien la pression artérielle qu'un état d'anxiété, et un vecteur unique doit écraser ces deux sens en un seul point. Pour trier correctement « le patient se plaint d'une tension à 19 » et « le patient paraît sous tension », il faut des représentations qui dépendent du contexte. C'est précisément ce que l'architecture suivante apporte.

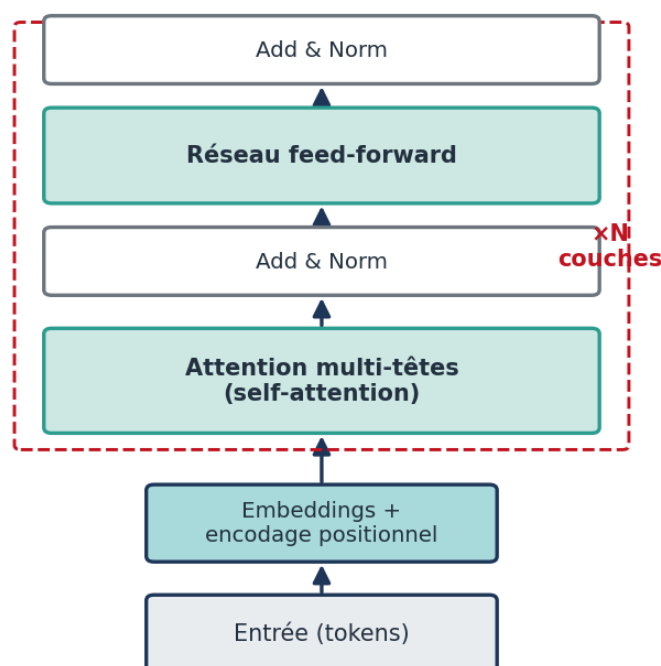
2.4.2 L'architecture Transformer

Avant 2017, les représentations contextuelles s'obtenaient avec des réseaux récurrents, qui lisent la phrase mot après mot en maintenant un état interne. Leur défaut est double : le traitement séquentiel interdit la parallélisation sur GPU, et l'information des premiers mots se dilue à mesure que la phrase s'allonge. L'article « Attention is all you need » de Vaswani et al. a éliminé la récurrence d'un coup, en construisant toute l'architecture autour d'un seul mécanisme : l'auto-attention [18].

Le principe se décrit bien en termes de requêtes et de réponses. Chaque mot de la séquence est projeté en trois vecteurs : une requête Q (ce que ce mot cherche dans son contexte), une clé K (ce que ce mot offre aux autres) et une valeur V (l'information qu'il transmet effectivement). L'attention calcule la compatibilité de chaque requête avec toutes les clés, la normalise en distribution de probabilité, et produit pour chaque mot une moyenne pondérée des valeurs :

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

Le facteur $\sqrt{d_k}$ stabilise simplement les produits scalaires quand la dimension des clés d_k

Architecture Transformer (bloc encodeur)**Fig. 2.3 :** Architecture Transformer (vue simplifiée).

grandit. Concrètement, dans « douleur abdominale apparue après le repas, sans fièvre », le mot « sans » apprend à porter une attention forte sur « fièvre » et en inverse la contribution, chose qu'un sac de mots ne pourra jamais faire. Chaque couche exécute plusieurs attentions en parallèle (attention multi-têtes), chacune libre de se spécialiser sur un type de relation différent, puis fait passer le résultat dans un petit réseau dense. L'empilement de ces blocs constitue le Transformer.

Un détail compte : l'attention étant une opération sur des ensembles, elle ignore l'ordre des mots. L'architecture y remédie par l'encodage positionnel, qui ajoute à l'embedding de chaque mot un vecteur dépendant de sa position dans la séquence, de sorte que « le patient a chuté » et « a chuté le patient » produisent des représentations distinctes [18]. Comme tous les mots sont traités simultanément, le Transformer exploite pleinement le parallélisme des GPU. C'est cette propriété qui a permis le passage à l'échelle des années suivantes. La figure 2.3 en donne une vue simplifiée.

2.4.3 Familles de modèles : encodeurs et décodeurs

Le Transformer original comportait deux moitiés, un encodeur et un décodeur, conçues pour la traduction automatique. La postérité les a dissociées, donnant naissance à deux familles aux vocations différentes.

Les modèles *encoder-only*, dont BERT est l'archétype [19], lisent la séquence entière dans

les deux sens : chaque mot voit tout son contexte, à gauche comme à droite. Pré-entraînés à deviner des mots masqués au milieu d'une phrase, ils excellent dans les tâches de compréhension (classification de textes, extraction d'entités, recherche sémantique) mais ne savent pas générer de texte. Les modèles *decoder-only*, lignée de GPT [20] poursuivie par LLaMA [21] et Mistral [22], font le choix inverse : chaque mot ne voit que ce qui le précède, et le modèle est entraîné à prédire le mot suivant. Cette contrainte, dite causale, est exactement ce qui rend la génération possible : produire un texte, c'est prédire le mot suivant en boucle. Le tableau 2.3 résume l'opposition.

Tab. 2.3 : Deux familles de modèles issues du Transformer.

	Encodeurs (BERT)	Décodeurs (GPT, LLaMA, Mistral)
Sens de lecture	bidirectionnel	causal (gauche → droite)
Pré-entraînement	mots masqués	prédiction du mot suivant
Forces	compréhension, classification	génération, dialogue, formats libres
Sortie	représentations, étiquettes	texte généré token par token

Notre cas d'usage tranche de lui-même. Un système de triage qui ne produirait qu'une étiquette parmi cinq pourrait se contenter d'un encodeur ; mais nous attendons du modèle une sortie riche (niveau, hypothèses diagnostiques, examens recommandés, justification) qu'il faut générer. C'est donc la famille des décodeurs qui s'impose, et c'est d'elle que dérive le modèle médical que nous présenterons en 2.5.5.

2.4.4 Tokenisation et embeddings

Une dernière étape sépare le texte brut du réseau : la tokenisation. Les modèles modernes ne travaillent ni au caractère (séquences trop longues) ni au mot entier (vocabulaire infini, mots inconnus), mais en sous-mots. L'algorithme BPE (*byte pair encoding*) construit le vocabulaire de façon gloutonne : partant des caractères, il fusionne itérativement les paires de symboles les plus fréquentes du corpus jusqu'à atteindre la taille de vocabulaire visée [16]. Les mots courants finissent par exister comme tokens entiers ; les mots rares sont décomposés en fragments, ce qui garantit que n'importe quelle chaîne, y compris un mot jamais vu, reste représentable.

Cette mécanique a une conséquence très concrète pour nous. Le vocabulaire de Mistral, environ 32 000 tokens, a été construit sur des corpus majoritairement anglophones [22] ; le vocabulaire médical français y est donc rarement présent en un seul morceau. Un terme comme « épigastrique » ou « hémoptysie » sera découpé en plusieurs sous-tokens, là où un mot anglais courant tient en un seul. Nos descriptions cliniques en français consomment ainsi davantage de tokens qu'un texte anglais équivalent, ce qui allonge les séquences à traiter, un paramètre à garder

en tête tant pour le coût d'entraînement que pour la latence d'inférence. Une fois la séquence tokenisée, chaque token est converti en vecteur dense par une simple table d'embeddings apprise avec le reste du modèle ; c'est ce flot de vecteurs, enrichi de l'encodage positionnel, qui entre dans la pile de blocs Transformer.

2.5 Grands modèles de langage

2.5.1 Pré-entraînement et capacités émergentes

Un grand modèle de langage (LLM, pour *large language model*) est un Transformer décodeur porté à très grande échelle : des dizaines de couches empilées, des matrices de poids géantes, et surtout un pré-entraînement sur des centaines de milliards de tokens de texte. La tâche de pré-entraînement n'a pourtant rien de sophistiqué : prédire le token suivant, encore et toujours. Mais résolue à cette échelle, elle force le modèle à absorber bien plus que la grammaire. Pour compléter correctement « devant une douleur thoracique avec sus-décalage du segment ST, le diagnostic à évoquer est », il ne suffit pas de connaître le français : il faut de la cardiologie. La connaissance s'installe dans les poids comme un sous-produit de la prédiction.

GPT-3 a montré ce que cette accumulation produit à 175 milliards de paramètres [20]. Brown et al. y documentent l'apprentissage en contexte (*in-context learning*) : le modèle accomplit des tâches pour lesquelles il n'a jamais été entraîné explicitement, simplement parce que son invite contient quelques exemples (*few-shot*), voire une consigne seule (*zero-shot*). Traduction, résumé, réponse à des questions, arithmétique élémentaire : ces capacités, dites émergentes parce qu'elles apparaissent avec la montée en échelle sans avoir été programmées, ont déplacé la frontière de ce qu'on peut attendre d'un modèle de langage.

Cette propriété intéresse directement le triage. Un LLM pré-entraîné arrive avec la langue, une capacité de raisonnement et une part du savoir médical déjà en place ; il sait lire « dyspnée brutale chez une patiente sous contraceptif oral » et y reconnaître un contexte évocateur d'embolie. Le travail d'adaptation ne part donc pas de zéro : il consiste à canaliser des capacités existantes vers une tâche précise, et c'est exactement la démarche que nous suivrons.

2.5.2 Du modèle généraliste au modèle spécialisé

Un modèle brut issu du pré-entraînement n'est cependant pas un assistant : c'est un compléteur de texte. Interrogé par « quel est le niveau de triage de ce patient ? », il peut tout aussi bien enchaîner sur une deuxième question, parce que des pages entières de son corpus ressemblaient à des listes de questions. L'ajustement par instructions (*instruction tuning*) corrige ce comportement : on poursuit l'entraînement sur des paires (consigne, réponse attendue) couvrant un large éventail de tâches, jusqu'à ce que le modèle prenne le pli de répondre à la consigne plutôt que de la prolonger. La plupart des modèles ouverts récents existent ainsi en deux versions, le modèle de

base et sa variante « instruct ».

La spécialisation à un domaine emprunte deux voies complémentaires. La première agit sur les connaissances : on poursuit le pré-entraînement sur des corpus du domaine (littérature biomédicale, comptes rendus cliniques) pour renforcer le vocabulaire et le savoir spécialisés ; c'est le pré-entraînement continué. La seconde agit sur la tâche : le fine-tuning supervisé ajuste le modèle sur des exemples de la tâche cible exacte, formatés comme à l'usage, soit dans notre cas un dossier d'admission en entrée, une évaluation de triage structurée en sortie. Les deux voies se cumulent sans difficulté, et c'est précisément notre stratégie : partir d'un modèle déjà pré-entraîné sur la littérature médicale, puis le spécialiser par fine-tuning supervisé sur notre corpus de cas de triage.

2.5.3 Quantification : fp16, INT8, INT4 et GGUF

Un modèle de 7,24 milliards de paramètres stockés en demi-précision (fp16, deux octets par poids) occupe une quinzaine de gigaoctets, avant même de compter le cache d'attention et les activations. Servir un tel modèle sur un matériel raisonnable impose de réduire cette empreinte, et la quantification en est l'outil principal : représenter chaque poids sur moins de bits, en acceptant une perte de précision numérique contrôlée. L'idée précède largement les LLMs, puisque la compression de réseaux par quantification était déjà étudiée au milieu des années 2010 [26], mais c'est avec eux qu'elle est devenue une étape obligée du déploiement. Le tableau 2.4 résume les formats courants.

Tab. 2.4 : Formats de précision numérique et empreinte mémoire approximative pour un modèle de 7 milliards de paramètres.

Format	Bits par poids	Taille (modèle 7 Mds)	Usage typique
FP32	32	≈ 29 Go	entraînement (référence historique)
fp16 / bf16	16	empreinte de référence	entraînement et inférence sur GPU
INT8	8	≈ 7 Go	inférence, perte de qualité minimale
INT4	4	≈ 4 Go	inférence sous forte contrainte mémoire

L'écosystème llama.cpp a standardisé cette pratique autour du format de fichier GGUF, qui regroupe en un seul fichier les poids quantifiés, le tokenizer et les métadonnées, et permet l'exécution du modèle sur GPU comme sur CPU [37]. Parmi ses schémas de quantification, Q4_K_M code les poids sur 4 bits par petits blocs munis de facteurs d'échelle, en réservant une

précision supérieure aux tenseurs les plus sensibles. Ce compromis qualité/taille est devenu un standard de fait, et c'est celui que nous avons retenu pour le déploiement : il réduit fortement l'empreinte du modèle sans dégradation perceptible sur notre tâche.

2.5.4 Inférence locale ou cloud

Reste à décider où ce modèle s'exécute. La voie du cloud consiste à appeler l'interface de programmation d'un fournisseur : elle donne accès aux modèles les plus puissants sans aucune infrastructure propre, mais au prix de trois dépendances. Un coût récurrent, facturé au volume de tokens et donc proportionnel à l'activité du service. Une latence et une disponibilité suspendues à la connexion internet, alors qu'un service d'urgences fonctionne la nuit, les jours de coupure réseau et les jours d'affluence. Et surtout, l'envoi de chaque requête vers des serveurs tiers, le plus souvent hébergés hors du territoire national.

Pour des données de santé, ce dernier point n'est pas un inconvénient parmi d'autres : il est disqualifiant. Une requête de triage contient l'âge, les constantes vitales, les antécédents et la plainte d'un patient identifiable par recoupement, soit des données parmi les plus sensibles qui soient. Les faire transiter par un cloud étranger soulève des questions de confidentialité, de conformité réglementaire et, plus fondamentalement, de souveraineté : un hôpital public ne devrait pas suspendre son processus de triage à la politique tarifaire et à la disponibilité d'un prestataire échappant à toute juridiction nationale.

L'inférence locale renverse ces termes. Le modèle s'exécute sur l'infrastructure de l'établissement ; aucune donnée ne sort ; le coût se réduit à l'amortissement du matériel ; le système continue de fonctionner si la connexion tombe. La contrepartie classique, à savoir qu'il faut un modèle assez compact pour le matériel disponible, est précisément ce que la quantification vient de résoudre. Notre choix était donc arrêté avant même d'être discuté : BetterEMS exécute son modèle localement, et nous verrons qu'un unique serveur GPU lui suffit pour évaluer un patient en deux à quatre secondes.

2.5.5 LLMs médicaux

L'aptitude des LLMs au raisonnement médical a été établie de façon retentissante par Med-PaLM, présenté par Singhal et al. dans la revue *Nature* [36]. Adapté d'un modèle généraliste de très grande taille, Med-PaLM atteint sur des bancs d'essai de questions médicales (dont des questions du type de l'examen de licence médicale américain) des performances que les auteurs situent au niveau d'experts humains, démontrant qu'un modèle de langage peut encoder une connaissance clinique réelle et la restituer avec cohérence. La démonstration a toutefois ses limites pour qui veut construire un système : Med-PaLM est propriétaire, immense, et accessible uniquement à travers l'infrastructure de son éditeur, donc inutilisable pour un déploiement local souverain.

La communauté ouverte a produit ses propres modèles médicaux. Meditron [44] poursuit le pré-entraînement de Llama-2 sur la littérature médicale et compte parmi les modèles ouverts les mieux évalués sur les bancs d'essai biomédicaux. MedAlpaca [45], plus modeste, ajuste LLaMA par instructions sur des questions-réponses médicales. L'un comme l'autre restent presque exclusivement anglophones.

BioMistral occupe une place à part dans ce paysage, et c'est lui que nous avons retenu [23]. Proposé par une équipe de recherche française, il dérive de Mistral-7B [22] par pré-entraînement continué sur PubMed Central, l'archive ouverte de la littérature biomédicale. Trois caractéristiques en font la base qu'il nous fallait. Il est médical : le pré-entraînement continué lui a donné le vocabulaire et les connaissances du domaine, là où un Mistral généraliste devrait tout réapprendre. Il est multilingue, évalué par ses auteurs sur des bancs d'essai médicaux traduits en plusieurs langues dont le français, la langue de nos données, de nos utilisateurs et de nos futurs cas réels. Il est enfin open-source, avec des poids librement téléchargeables et une taille de 7 milliards de paramètres compatible avec un fine-tuning sur le GPU dont nous disposons. C'est sur cette base que nous construirons, par adaptation sur notre corpus propriétaire, notre modèle spécialisé BetterEMS-Triage de 7,24 milliards de paramètres ; les chapitres 3 et 4 en détaillent la fabrication.

2.6 Adapter un LLM : le fine-tuning efficace

2.6.1 Transfer learning et PEFT

Adapter un modèle pré-entraîné plutôt que d'entraîner depuis zéro relève du transfert d'apprentissage : les représentations acquises sur un corpus immense servent de point de départ, et seul l'ajustement final reste à accomplir. La méthode directe, le fine-tuning complet, reprend l'entraînement sur les nouvelles données en mettant à jour la totalité des paramètres. Elle se heurte à un mur mémoire. Pour chaque paramètre entraîné, il faut conserver le poids lui-même, son gradient et les états internes de l'optimiseur, sachant qu'AdamW entretient deux moyennes mobiles par paramètre. Selon les précisions retenues pour chacun de ces tenseurs, la facture atteint l'ordre de la centaine de gigaoctets pour un modèle de 7 milliards de paramètres : plus de quatre fois la capacité de notre serveur GPU du CRI, et hors de portée d'une carte grand public prise isolément.

Les méthodes de fine-tuning économe en paramètres (PEFT, *parameter-efficient fine-tuning*) contournent l'obstacle en gelant le modèle de base et en n'entraînant qu'un petit ensemble de paramètres additionnels ; la bibliothèque PEFT de Hugging Face en fournit les implémentations de référence [40]. Les poids gelés n'ont besoin ni de gradients ni d'états d'optimiseur, et la mémoire d'entraînement chute en proportion. Le pari sous-jacent est que l'adaptation à une tâche ne nécessite pas de réécrire tout le modèle. La pratique l'a très largement confirmé, et sa formalisation la plus aboutie est LoRA.

Principe de LoRA : adaptation par matrices de rang faible

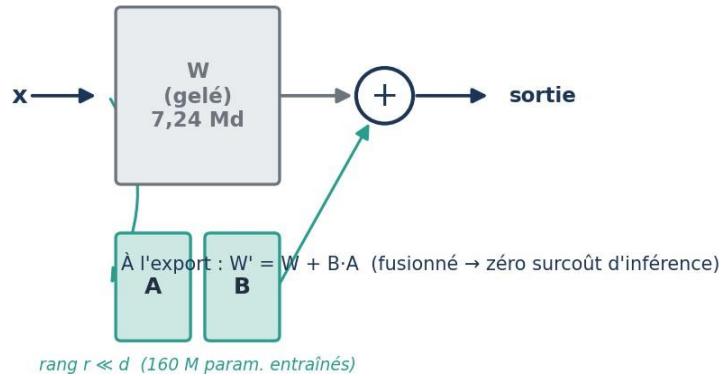


Fig. 2.4 : Principe de LoRA : matrices basse-rang A et B en parallèle du poids gelé W.

2.6.2 LoRA

L'observation fondatrice de LoRA (*Low-Rank Adaptation*) est que la modification des poids induite par une adaptation possède un rang intrinsèque faible : la mise à jour utile vit dans un sous-espace de dimension très réduite [24]. Plutôt que d'apprendre directement une correction de même taille que la matrice d'origine $W \in \mathbb{R}^{d \times k}$, LoRA la factorise en un produit de deux matrices étroites :

$$W^t = W + BA$$

avec $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$ et un rang $r \ll \min(d, k)$. Seules A et B sont entraînées ; W reste gelée. Le gain est massif. Pour une matrice 4096×4096 , soit l'ordre de grandeur des projections d'attention de notre modèle, et un rang $r = 32$, les deux matrices basse-rang totalisent environ 262 000 paramètres, contre 16,8 millions pour la matrice pleine : moins de 2 %. La figure 2.4 illustre le montage, les deux matrices opérant en parallèle du poids gelé. Appliqué à l'ensemble des projections d'attention et des couches denses de notre modèle, ce dispositif ramène les paramètres entraînaibles à environ 84 millions, soit 1,1 % des 7,24 milliards.

LoRA possède une seconde vertu, décisive pour la mise en production : à l'issue de l'entraînement, le produit BA peut être fusionné dans W une fois pour toutes. Le modèle exporté est alors une matrice unique $W^t = W + BA$, structurellement identique au modèle d'origine. L'inférence ne porte aucun surcoût, ni en calcul ni en mémoire, et toute la chaîne d'outils standard, quantification comprise, s'applique telle quelle au modèle fusionné.

2.6.3 QLoRA et ses arbitrages

QLoRA pousse l'économie un cran plus loin : puisque les poids de base sont gelés, pourquoi les conserver en pleine précision [25] ? La méthode les quantifie en NF4 (*NormalFloat 4 bits*), un format dont les seize niveaux sont calés sur la distribution réellement observée des poids des réseaux, y ajoute une double quantification des facteurs d'échelle, et ne maintient en précision élevée que les adaptateurs LoRA et leurs gradients. Dettmers et al. montrent qu'on peut ainsi affiner des modèles de plusieurs dizaines de milliards de paramètres sur une seule carte, avec des résultats comparables au fine-tuning 16 bits dans leurs conditions d'évaluation [25].

L'arbitrage n'est cependant pas gratuit. La quantification injecte un bruit systématique dans les poids gelés, donc dans tous les calculs que traversent les gradients des adaptateurs. Sur de grands jeux de données variés, ce bruit se moyenne et s'oublie ; sur un petit corpus spécialisé, où chaque pas de gradient porte un signal ténu, il peut suffire à déstabiliser l'optimisation : pertes qui oscillent, voire qui divergent. Le tableau 2.5 résume la balance entre les deux méthodes. Nous verrons au chapitre 4 que cet arbitrage n'a, pour nous, rien eu de théorique : nos deux campagnes d'entraînement l'ont mis à l'épreuve directement, et c'est lui qui les a départagées.

Tab. 2.5 : LoRA et QLoRA : termes de l'arbitrage entre mémoire et stabilité.

Critère	LoRA (base fp16)	QLoRA (base NF4)
Précision des poids gelés	16 bits	4 bits (NF4)
Mémoire GPU (modèle 7 Mds)	élevée (le modèle de base reste en pleine précision)	réduite, GPU grand public envisageable
Stabilité de l'optimisation	éprouvée	sensible sur les petits corpus spécialisés
Qualité finale rapportée	référence	comparable dans les conditions favorables [25]

2.6.4 Prompt engineering et sorties structurées

Les poids ne sont pas le seul levier : le comportement d'un LLM se façonne aussi par son invite. Le prompt système, un texte fixe antéposé à chaque requête, définit le rôle du modèle, ses contraintes et le format attendu de sa réponse ; le message utilisateur porte les données du cas à traiter. Bien rédigé, un prompt système suffit à transformer un modèle généraliste en évaluateur de triage passable ; il ne suffit pas à en faire un composant logiciel.

Car pour brancher un LLM sur une API métier, une sortie en texte libre est inutilisable : un programme n'interprète pas une réponse rédigée, il l'analyse champ par champ. La solution standard consiste à imposer une sortie structurée, en pratique un objet JSON au schéma fixé. Notre modèle doit ainsi répondre par un objet unique dont les champs (niveau et couleur Manchester, confiance, score, délai d'attente maximal, hypothèses diagnostiques avec codes CIM-10 et probabilités,

examens recommandés, précautions, justification, indicateurs d'alerte) sont consommés tels quels par le backend du système. Obtenir d'un modèle de 7 milliards de paramètres qu'il respecte un tel schéma par la seule force de l'invite reste aléatoire ; c'est l'une des raisons de notre choix du fine-tuning. Entraîné sur des milliers d'exemples rédigés exactement dans ce format, le modèle l'adopte comme sa langue de sortie naturelle, et la validation du JSON à la réception ne sert plus que de filet de sécurité.

2.7 L'IA appliquée au triage des urgences

2.7.1 Panorama des travaux

L'application de l'apprentissage automatique au triage des urgences s'est structurée au cours de la dernière décennie, d'abord sur données tabulaires. Levin et al. ont développé *e-triage*, un système fondé sur des forêts aléatoires entraînées sur les données recueillies en routine à l'admission (constantes, motif de recours, antécédents codés), et l'ont confronté à l'ESI sur de grandes cohortes rétrospectives américaines [32]. Les auteurs rapportent une discrimination des issues critiques supérieure à celle de l'ESI, en particulier au sein du niveau intermédiaire ESI 3, où l'échelle d'origine range côte à côte des patients aux risques en réalité très différents.

D'autres équipes ont déplacé la cible : plutôt que de reproduire l'étiquette de triage, prédire directement les issues cliniques. Raita et al. entraînent quatre familles de modèles (régression pénalisée, forêts aléatoires, gradient boosting, réseau de neurones) sur les données d'une enquête nationale américaine, pour prédire l'hospitalisation et le recours aux soins critiques dès le triage ; ils rapportent une discrimination supérieure à celle du triage de référence [33]. Kwon et al. valident quant à eux un score d'acuité par apprentissage profond sur une très grande base nationale coréenne de passages aux urgences ; selon les auteurs, le score prédit la mortalité hospitalière mieux que les outils en usage, et la taille de la base démontre que l'approche passe à l'échelle d'un système de santé entier [34].

Ces travaux partagent une limite : ils n'exploitent que des variables structurées, alors que l'information la plus riche du triage, la plainte du patient, arrive en texte libre. Ivanov et al. franchissent ce pas en combinant apprentissage automatique et traitement du langage naturel clinique sur les notes infirmières pour l'attribution du niveau ESI ; sur leurs données rétrospectives multi-établissements, les auteurs rapportent une exactitude supérieure à celle du triage pratiqué, l'apport du texte libre étant précisément ce qui fait la différence [35].

Les grands modèles de langage prolongent ce mouvement. Med-PaLM, déjà évoqué, n'est pas un système de triage, mais il établit qu'un LLM peut porter une connaissance clinique de niveau expert et la mobiliser sur des cas décrits en langage naturel [36]. La brique technologique existe ; ce qui manque, c'est son assemblage en un système de triage opérationnel. C'est très exactement l'objet de notre travail.

2.7.2 Tableau comparatif

Le tableau 2.6 met ces travaux en regard, notre système figurant en dernière ligne.

Tab. 2.6 : Travaux représentatifs sur le triage assisté par IA et positionnement de BetterEMS.

Travaux	Année	Approche	Données	Apport	Limites
Levin et al. [32]	2018	Forêts aléatoires (e-triage)	Cohortes rétrospectives, États-Unis	Discrimination supérieure à l’ESI ; stratification du niveau 3	Variables structurées seules ; anglais ; pas de justification clinique
Kwon et al. [34]	2018	Score d’acuité par apprentissage profond	Très grande base nationale, Corée du Sud	Validation à l’échelle d’un système de santé	Données structurées ; score sans explication
Raita et al. [33]	2019	Régression, forêts, gradient boosting, réseau de neurones	Enquête nationale, États-Unis	Prédiction d’issues cliniques dès le triage	Tabulaire ; étude rétrospective sans déploiement
Ivanov et al. [35]	2021	ML + NLP clinique	Dossiers multi-établissements, États-Unis	Intégration du texte infirmier libre	Cible l’ESI ; anglais ; sortie limitée au niveau
Singhal et al. [36]	2023	LLM médical (Med-PaLM)	Bancs d’essai de questions médicales	Connaissance clinique de niveau expert dans un LLM	Non spécialisé triage ; propriétaire ; déploiement local impossible

Travaux	Année	Approche	Données	Apport	Limites
BetterEMS (notre travail)	2026	LLM de 7,24 milliards de paramètres, fine-tuning LoRA	Corpus propriétaire de 12 400 cas cliniques en français	Sortie structurée Manchester complète ; système de bout en bout déployé en production	Validation clinique programmée

Deux lignes de fracture traversent ce tableau. La première sépare les approches selon les données qu’elles savent lire : quatre travaux sur six se limitent aux variables structurées, et seul Ivanov et al. exploitent le texte clinique, sans toutefois le générer ni le restituer. La seconde sépare les études des systèmes : tous ces travaux sont des évaluations rétrospectives ou des bancs d’essai ; aucun ne décrit un système complet en service dans un flux de patients. BetterEMS est le seul dispositif du tableau à couvrir la chaîne entière, de la saisie à l’écran de salle d’attente.

2.7.3 Lacunes et positionnement

La lecture transversale de ces travaux fait ressortir trois lacunes nettes. La première est linguistique et géographique : la littérature du triage assisté par IA est presque entièrement anglophone, construite sur des cohortes nord-américaines ou est-asiatiques et adossée à l’ESI. Les travaux en français sont rarissimes, et nous n’avons identifié aucun système conçu pour le contexte maghrébin : son épidémiologie, ses libellés cliniques en français, son organisation hospitalière. La deuxième est une lacune de représentation : le tabulaire domine, alors même que l’expérience d’Ivanov et al. montre que le texte libre porte un signal décisif [35] ; les systèmes capables de lire une plainte rédigée, et a fortiori d’y répondre par une évaluation argumentée, restent l’exception. La troisième est une lacune de réalisation : la quasi-totalité des contributions s’arrête à l’étude rétrospective, et les systèmes déployés de bout en bout (du formulaire d’admission à la file d’attente pilotée) sont à peu près absents de la littérature.

BetterEMS s’est construit sur ces trois axes à la fois. Notre corpus de 12 400 cas est rédigé en français et conçu pour le contexte algérien, pathologies locales comprises ; notre modèle lit conjointement les constantes vitales et la description textuelle des symptômes, et restitue une évaluation Manchester structurée et justifiée ; et le système qui l’entoure est complet et déployé : enregistrement, constantes, triage, file temps réel, validation médicale. Aucun des travaux examinés ne réunit ces trois propriétés ; c’est ce qui fonde, à notre connaissance, l’originalité de notre contribution.

2.8 Conclusion

Ce chapitre a déroulé la chaîne complète qui mène du problème posé au chapitre 1 aux outils capables de le résoudre. Le triage de Manchester s'est laissé formuler en classification supervisée à cinq classes, avec le rappel des classes graves pour métrique cardinale ; les réseaux de neurones, du perceptron de 339 paramètres au Transformer, ont fourni la mécanique d'apprentissage ; les grands modèles de langage y ont ajouté la lecture du texte clinique libre, et BioMistral la connaissance médicale en français. Les techniques d'adaptation transforment ces capacités en composants d'ingénierie : LoRA pour entraîner 1,1 % des paramètres, la quantification GGUF pour servir le modèle en local, les sorties JSON pour le brancher sur une application. La revue des travaux existants a enfin situé notre projet : un système de triage en français, lisant le texte libre, déployé de bout en bout, configuration qu'aucun travail recensé ne propose. Tous les éléments sont désormais en place. Le chapitre suivant présente la conception et la mise en œuvre de BetterEMS : son architecture, son corpus, son modèle et son déploiement.

Conception et mise en œuvre du système BetterEMS

3.1 Introduction

Les deux chapitres précédents ont délimité le problème et les outils : d'un côté, un service d'urgences où la file d'attente doit être réordonnée selon les cinq niveaux du protocole de Manchester ; de l'autre, des modèles de langage que l'on sait désormais adapter à un domaine précis avec des moyens de calcul raisonnables. Ce chapitre décrit ce que nous avons construit à la jonction des deux.

Notre parti pris mérite d'être énoncé d'emblée : BetterEMS n'est pas un prototype de laboratoire qui classerait des cas dans un notebook. C'est un système hospitalier complet, pensé pour être utilisé debout, dans le bruit d'une salle d'attente, par cinq catégories de personnel différentes. Le modèle d'intelligence artificielle n'en est qu'une composante parmi d'autres, certes centrale, mais encadrée par une application métier, une file d'attente temps réel et des mécanismes de sécurité explicites. Cette exigence de bout en bout a orienté chacun de nos choix : un modèle inutilisable hors connexion, trop lent à répondre ou impossible à auditer aurait été éliminé d'office, quelles que soient ses performances brutes.

La démarche suit l'ordre naturel de construction. Nous partons des besoins du terrain et des acteurs (3.2), puis nous posons l'architecture logicielle (3.3) et son déploiement physique (3.4). Vient ensuite la matière première du modèle : notre corpus propriétaire de cas de triage, de sa rédaction manuelle à son découpage en ensembles d'entraînement et d'évaluation (3.5). Sur cette base, nous présentons notre modèle BetterEMS-Triage, de son modèle de fondation à son export en format de production (3.6), puis son intégration dans le système avec les garde-fous qui l'accompagnent (3.7) et la boucle d'apprentissage continu (3.8). Les résultats expérimentaux, à savoir les courbes d'entraînement, la précision mesurée et la vitesse d'inférence, font l'objet du chapitre 4 ; nous y renverrons chaque fois qu'un choix de conception appelle une validation chiffrée.

3.2 Analyse des besoins et acteurs

Avant d'écrire la moindre ligne de code, nous avons reconstitué le parcours réel d'un patient aux urgences, tel qu'il se déroule dans un hôpital algérien. Ce parcours fait intervenir des métiers distincts, aux responsabilités bien séparées, et le logiciel devait épouser cette séparation plutôt que la redessiner : un outil qui oblige le personnel à changer ses habitudes de travail finit débranché.

Nous avons retenu cinq rôles, qui correspondent aux acteurs effectivement présents dans la chaîne de prise en charge.

Tab. 3.1 : Les cinq rôles du système BetterEMS et leurs responsabilités.

Rôle	Responsabilités dans le système
Agent d'accueil	Enregistre l'identité du patient, ouvre la visite, imprime le ticket
Infirmier	Saisit les constantes vitales et les symptômes, déclenche le triage
Médecin	Consulte la file priorisée, appelle le patient en box, confirme ou corrige le triage
Ambulancier (SAMU)	Pré-enregistre un patient pendant le transport vers l'hôpital
Administrateur	Gère les comptes utilisateurs, les paramètres du service et les statistiques

Le parcours nominal s'enchaîne comme suit. Le patient se présente à l'accueil ; l'agent saisit son identité, ouvre une visite et lui remet un ticket numéroté. L'infirmier le reçoit ensuite pour relever ses constantes vitales (température, tension, fréquences cardiaque et respiratoire, saturation en oxygène) et consigner ses symptômes, la localisation et l'intensité de sa douleur. Ces données déclenchent le triage : le moteur d'intelligence artificielle propose un niveau de Manchester, accompagné d'hypothèses diagnostiques et d'une justification. Le patient est alors inséré dans la file d'attente à la position que son niveau lui confère, et non à celle de son ordre d'arrivée. Quand son tour vient, l'écran de la salle d'attente affiche son ticket et le box où se rendre ; le médecin l'examine, puis confirme ou corrige la proposition de triage. C'est sa décision, et elle seule, qui fait foi. Le cas du SAMU s'y greffe naturellement : l'ambulancier pré-enregistre le patient pendant le transport, si bien que le dossier existe déjà quand l'ambulance franchit le portail.

La figure 3.1 résume les cas d'utilisation du système et l'acteur associé à chacun.

L'enchaînement temporel de ces interactions, du relevé des constantes à la validation par le médecin, est détaillé par le diagramme de séquence de la figure 3.2.

Le ticket imprimé peut sembler anachronique dans un système temps réel ; il est en réalité indispensable. C'est l'identifiant que le patient garde en main, celui que l'écran de la salle d'attente affichera au moment de l'appel, et le seul lien entre une personne dans la foule et son dossier numérique. À l'autre bout de la chaîne, l'administrateur porte un besoin moins visible mais bien réel : le pilotage du service. Temps d'attente par niveau, volume de passages, taux d'accord entre le médecin et l'IA : ces indicateurs, à peu près inexistant dans un service fonctionnant au papier, sont produits par BetterEMS en sous-produit de son fonctionnement normal.

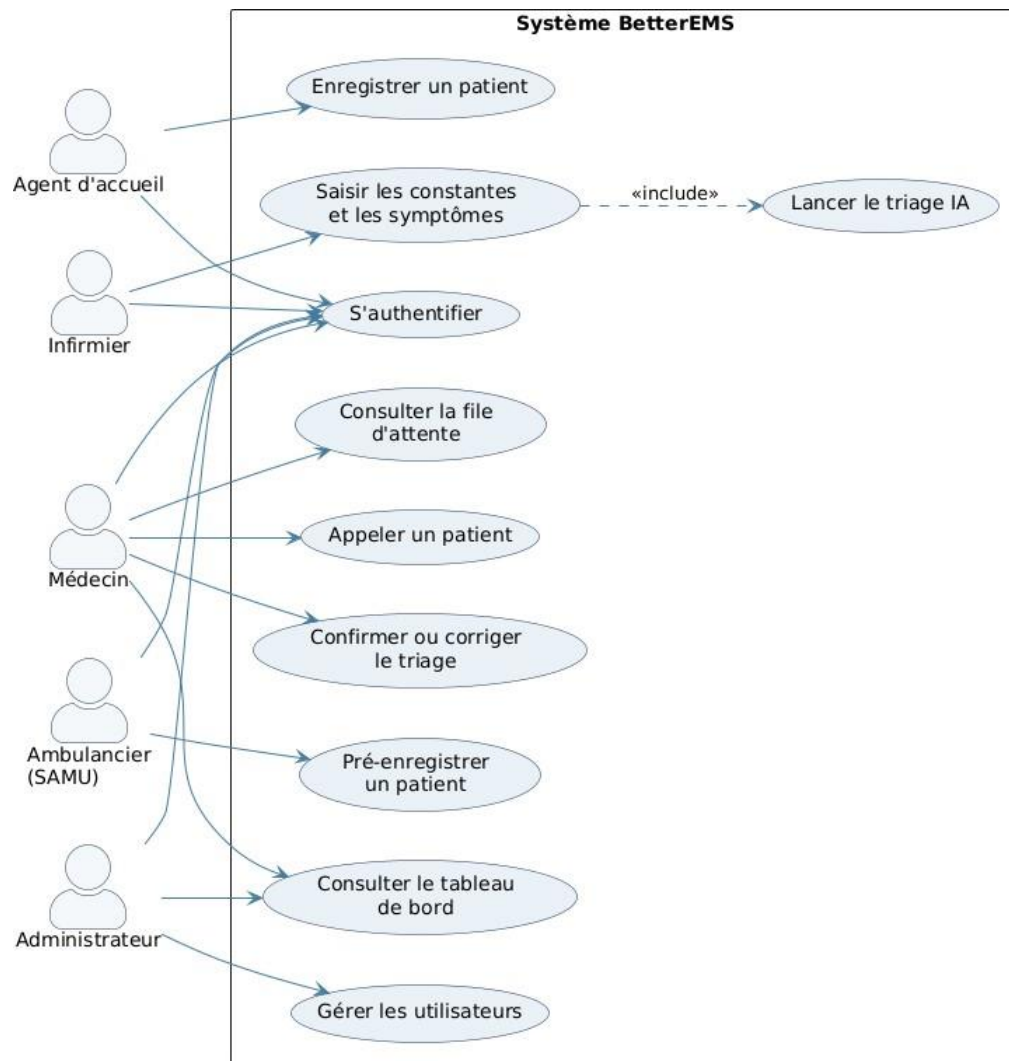


Fig. 3.1 : Diagramme de cas d'utilisation de BetterEMS.

De cette analyse découlent trois exigences qui ont structuré toute la conception. La première est le temps de réponse : l'évaluation automatique doit tenir en quelques secondes, faute de quoi elle ralentirait l'accueil au lieu de le fluidifier, un comble pour un outil censé absorber les pics d'affluence. La deuxième est la validation humaine systématique : aucune orientation de patient ne se fait sur la seule sortie du modèle ; le médecin garde le dernier mot sur chaque triage, conformément aux recommandations internationales sur l'IA en santé [30]. La troisième est le fonctionnement entièrement local : les données de santé ne quittent pas l'infrastructure maîtrisée par l'établissement, ce qui exclut d'emblée tout recours à un modèle hébergé sur un cloud étranger. Ces trois contraintes ne sont pas des vœux pieux ajoutés en fin de projet ; elles ont éliminé des options techniques entières, comme nous le verrons au moment du choix du modèle (3.6.1).

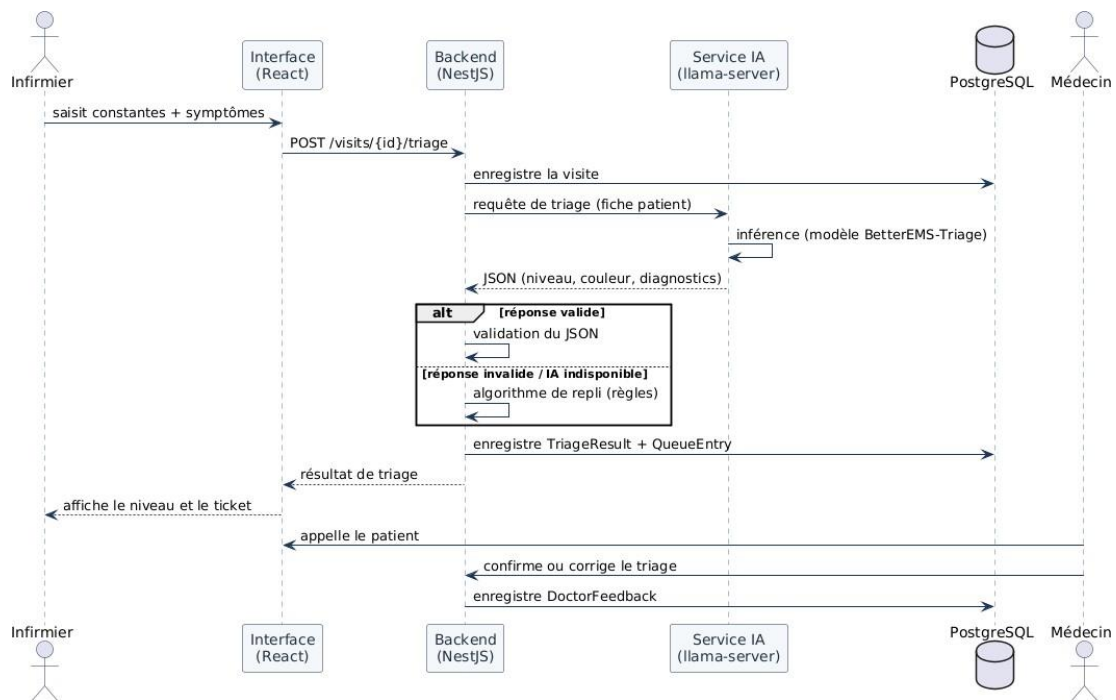


Fig. 3.2 : Diagramme de séquence du processus de triage.

3.3 Architecture globale

L'architecture répond à un découpage classique en couches, mais avec une particularité : le moteur d'intelligence artificielle est un service à part entière, isolé du reste de l'application, que le backend interroge comme il interrogerait n'importe quelle API externe. Cette isolation n'est pas un détail d'implémentation. Elle permet de mettre à jour le modèle sans toucher à l'application, de le faire tourner sur une machine différente, ce qui sera le cas en production, et surtout de prévoir sa défaillance : si le service ne répond pas, le reste du système continue de fonctionner. La figure 3.3 donne la vue d'ensemble.

Quatre composantes se partagent le travail. Le frontend, développé en React 19 et TypeScript [43] avec Tailwind v4 et Vite 7, regroupe onze pages : connexion, enregistrement des patients, saisie des constantes, triage, file d'attente, administration, tableaux de bord, et l'écran TV de la salle d'attente. Chaque rôle n'accède qu'aux pages qui le concernent. Le typage statique de TypeScript nous a évité une catégorie entière d'erreurs : les structures échangées avec le backend (résultat de triage, entrée de file) sont décrites par des types partagés, et une incohérence de champ se détecte à la compilation plutôt qu'en salle d'urgences.

Le backend, écrit avec le framework NestJS [42] et l'ORM TypeORM, expose l'API REST et concentre toute la logique métier. Il est organisé en modules qui recouvrent presque un pour un les besoins identifiés en 3.2 : authentification et contrôle d'accès (auth), gestion des patients (patients), visites avec constantes et symptômes (visits), triage et retours du médecin (triage), file d'attente avec sa passerelle temps réel (queue), indicateurs du service (dashboard), journalisation des actions (audit), paramètres (settings), notifications, et le module ai qui dialogue avec le

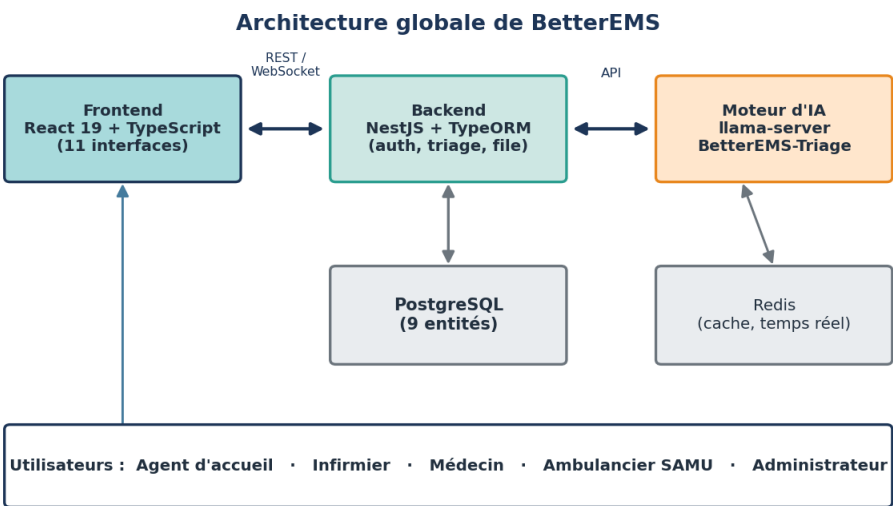


Fig. 3.3 : Architecture globale de BetterEMS.

moteur d’inférence. Deux canaux de communication coexistent : REST pour les opérations classiques de création et de consultation, et WebSocket, via Socket.io, pour tout ce qui doit se propager sans délai, au premier rang la file d’attente, dont chaque modification est poussée vers les écrans connectés.

Le moteur d’inférence héberge notre modèle BetterEMS-Triage derrière une API compatible OpenAI ; nous détaillons son fonctionnement en 3.7.1. La persistance, enfin, repose sur PostgreSQL [41], dont la robustesse transactionnelle convient à des données qui engagent la responsabilité de l’établissement. Le schéma relationnel compte neuf entités, que le tableau 3.2 résume.

Tab. 3.2 : Les neuf entités du modèle de données de BetterEMS.

Entité	Rôle
User	Compte d’un membre du personnel, avec son rôle et ses identifiants.
Patient	Identité et informations administratives d’un patient, conservées d’une visite à l’autre.
Visit	Un passage aux urgences, de l’enregistrement à la sortie, avec son statut courant.
VitalSigns	Un relevé de constantes vitales rattaché à une visite, horodaté.
Symptom	Un symptôme déclaré ou observé, avec sa localisation et le niveau de douleur associé.
TriageResult	La proposition du moteur de triage : niveau, couleur, confiance, diagnostics, examens, précautions, explication.

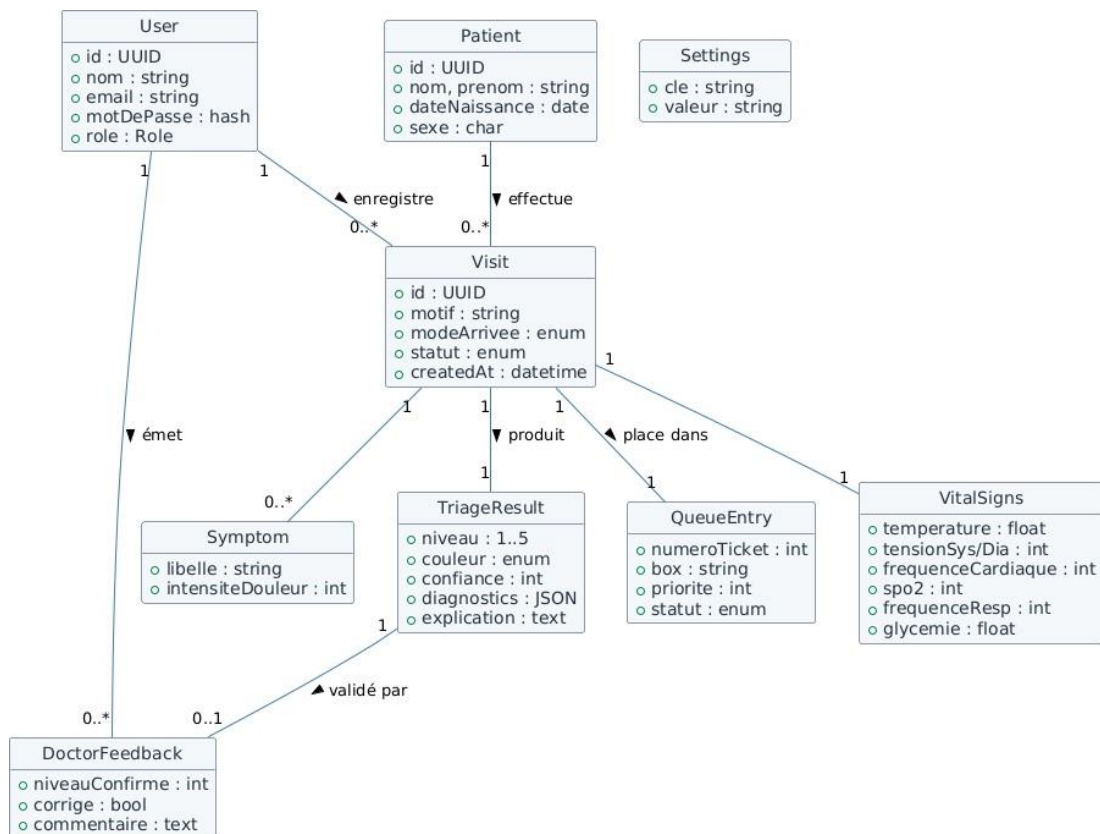


Fig. 3.4 : Diagramme de classes du modèle de données.

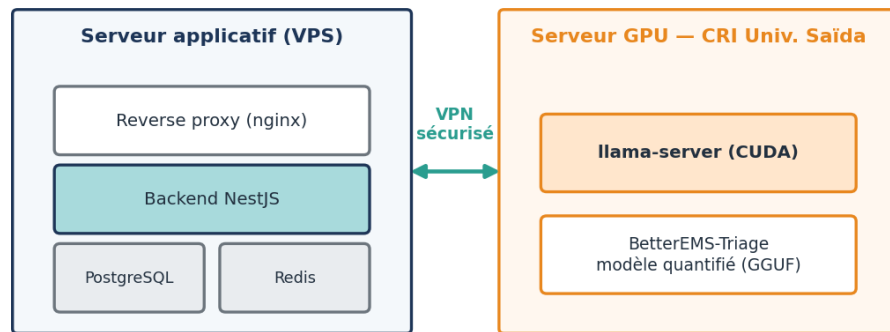
Entité	Rôle
DoctorFeedback	La décision du médecin sur cette proposition : confirmation, ou correction motivée.
QueueEntry	La position du patient dans la file priorisée, avec son ticket et le box d'appel.
Settings	Les paramètres configurables du service (boxes, seuils, affichage).

La figure 3.4 présente ces neuf entités et leurs associations sous forme de diagramme de classes UML.

On remarquera que TriageResult et DoctorFeedback sont deux entités distinctes. La proposition de la machine et la décision du médecin ne sont jamais confondues dans le même enregistrement : cette séparation, décidée dès la conception du schéma, est ce qui rend possible à la fois la traçabilité (qui a décidé quoi, et quand) et la boucle d'apprentissage continu décrite en 3.8.

Pour fixer les idées, suivons une demande de triage à travers ces couches. L'infirmier valide sa saisie ; le frontend adresse une requête REST au module visits, qui enregistre les constantes et les symptômes, puis le module triage prend la main. Celui-ci délègue au module ai la construction du prompt et l'appel du moteur d'inférence; la réponse, une fois validée, est persistée comme

Architecture de déploiement



Les données de santé restent sous le contrôle de l'établissement

Fig. 3.5 : Architecture de déploiement : VPS applicatif et serveur GPU reliés par un réseau privé virtuel sécurisé.

TriageResult, et le module queue crée l'entrée de file correspondante. La passerelle Socket.io diffuse aussitôt le nouvel état : le poste du médecin se met à jour, l'écran de la salle d'attente aussi. Du clic de l'infirmier à l'apparition du patient dans la file, il n'y a aucune intervention manuelle, et chaque étape laisse une trace dans le journal d'audit.

3.4 Architecture de déploiement

Un modèle de sept milliards de paramètres ne s'héberge pas sur le même serveur qu'une application web ordinaire : il lui faut un GPU. Plutôt que de renoncer au déploiement réel ou de louer du calcul à l'étranger, ce que l'exigence de localité des données interdisait, nous avons réparti le système sur deux machines aux profils complémentaires, reliées par un réseau privé virtuel (VPN) sécurisé. La figure 3.5 illustre ce découpage.

Le premier serveur, un VPS applicatif, porte tout ce qui ne demande pas de GPU : le reverse proxy (nginx) qui termine les connexions des clients et sert le frontend, le backend NestJS, la base PostgreSQL et Redis, ce dernier assurant le cache et le support des files temps réel. Le second est le serveur GPU du CRI (Centre de Recherche en Informatique) de l'Université de Saïda. C'est sur cette machine que notre modèle a été entraîné, et c'est elle qui assure l'inférence en production. Le VPN établit entre les deux un canal chiffré de point à point : les requêtes de triage transitent du backend vers le moteur d'inférence sans jamais s'exposer sur l'Internet public, et aucune donnée de patient ne touche un service tiers.

Ce partenariat avec le CRI a rendu le projet matériellement possible, et il faut le rappeler : l'accès à un GPU, pour l'entraînement comme pour le service, est rarement à la portée d'un projet de Master. Il préfigure aussi le schéma de déploiement cible en milieu hospitalier, à savoir un serveur applicatif léger dans l'établissement et un nœud de calcul mutualisé à proximité, que nous

reprendrons dans l'étude entrepreneuriale du chapitre 5. Côté exploitation, nous nous en tenons à l'essentiel : les services sont supervisés par un gestionnaire de processus qui les relance en cas d'arrêt, et les échanges avec les postes clients sont protégés par TLS. Le détail de la configuration n'a pas sa place dans un document public, et n'apporterait rien à la compréhension du système.

3.5 Constitution du corpus propriétaire

3.5.1 Création manuelle des cas

Adapter un modèle de langage au triage suppose des exemples de triage, et il n'en existait aucun jeu public utilisable pour notre cas : les bases hospitalières ouvertes comme MIMIC-IV sont en anglais, centrées sur la réanimation, et ne portent pas d'étiquettes Manchester [27]; aucune ressource francophone équivalente n'est disponible, encore moins une qui refléterait les pathologies vues dans un service d'urgences algérien. Nous avons donc pris la décision la plus coûteuse en temps du projet : rédiger nous-mêmes un corpus de 12 600 cas de triage, en français.

Les cas se répartissent en sept familles cliniques, calibrées pour couvrir le spectre d'un service d'urgences : cardiologie et pneumologie ; traumatologie et brûlures ; pédiatrie et obstétrique ; gastro-entérologie, urologie et intoxications ; neurologie et psychiatrie ; pathologies infectieuses et présentations propres au contexte algérien ; et des cas mixtes qui croisent plusieurs systèmes. La sixième famille a demandé un travail de documentation spécifique : on y trouve, entre autres, l'envenimation scorpionique (notre corpus comporte par exemple le cas d'un enfant de cinq ans piqué dans la région de Biskra, arrivé en ambulance avec troubles de conscience et détresse respiratoire, étiqueté Rouge) ou encore le coup de chaleur sévère, deux urgences que les jeux de données nord-américains ignorent à peu près complètement. La rédaction elle-même a été cadrée par le protocole : pour chaque cas, le niveau attendu découle d'un discriminant Manchester identifiable, et non d'une impression d'ensemble, ce qui rend chaque étiquette justifiable et vérifiable a posteriori [6]. Les cas mixtes, volontairement plus retors, croisent plusieurs tableaux cliniques chez un même patient, précisément pour empêcher le modèle d'apprendre des associations trop simples entre un mot-clé et un niveau.

Chaque cas suit une structure fixe. L'entrée décrit le patient tel que l'infirmier le verrait : âge, sexe, motif de consultation, liste des symptômes, niveau de douleur sur dix et ses localisations, constantes vitales complètes, antécédents, traitements en cours, allergies, mode d'arrivée, début et évolution des troubles. La sortie attendue donne la décision de triage et son contexte clinique : niveau et couleur Manchester, hypothèses diagnostiques codées en CIM-10 avec leur probabilité, examens complémentaires recommandés et précautions de prise en charge. Pour le codage diagnostique, nous nous sommes adossés à un dictionnaire d'environ 109 000 codes CIM-10 extrait de MIMIC-IV Demo [27] (et son prédécesseur MIMIC-III [28]). Précisons-le clairement : seuls les codes et leurs libellés en proviennent ; les cas cliniques eux-mêmes sont entièrement de notre main.

Ce caractère synthétique du corpus est un choix de conception, pas un pis-aller. Aucun cas

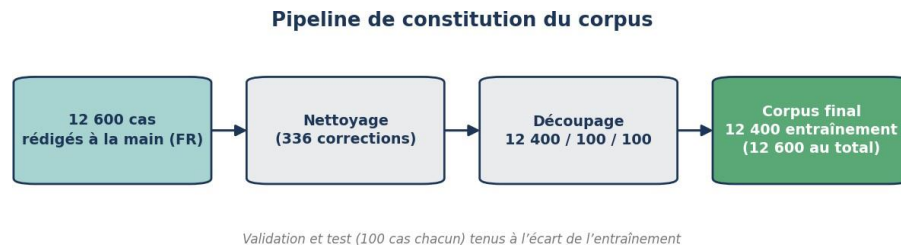


Fig. 3.6 : Constitution du corpus de triage (12 600 cas rédigés à la main, découpés en 12 400 / 100 / 100).

ne décrit un patient réel : il n’y a donc ni consentement à recueillir, ni anonymisation à garantir, ni autorisation réglementaire à obtenir pour manipuler, partager ou publier ces données, un avantage décisif quand on sait combien l’accès aux données de santé réelles est long et incertain. La contrepartie est connue : un corpus rédigé ne capture pas toute la variabilité du réel, et c’est précisément ce que la boucle d’apprentissage continu (3.8) viendra corriger avec les retours des médecins.

3.5.2 Nettoyage et contrôle qualité

Un corpus de cette taille rédigé à la main accumule inévitablement des incohérences. Nous avons écrit un script de vérification systématique (`fix_data.py`) qui a relevé puis corrigé 336 problèmes : constantes vitales manquantes dans certains relevés, noms de champs orthographiés différemment d’une famille de cas à l’autre, niveaux de douleur hors de l’échelle 0 à 10, et quelques doublons. L’opération peut sembler triviale ; elle ne l’est pas. Un modèle de langage apprend tout ce qu’on lui montre, y compris les défauts de format : un champ tantôt nommé d’une façon, tantôt d’une autre, se retrouverait dans ses sorties.

3.5.3 Découpage et distribution du corpus

Le corpus nettoyé a été découpé en trois ensembles disjoints, selon le protocole rappelé au chapitre 2 : 12 400 cas d’entraînement, 100 de validation, 100 de test, avec une graine aléatoire fixée à 42. Ce détail garantit la reproductibilité exacte du découpage : quiconque relance le script obtient les mêmes trois ensembles. L’ensemble de test a été mis sous clé dès ce moment ; il ne ressortira qu’au chapitre 4, pour l’évaluation finale. Au total, le corpus compte 12 600 cas (12 400 pour l’entraînement, 100 pour la validation, 100 pour le test) ; c’est sur ces 12 400 exemples que notre modèle BetterEMS-Triage a été entraîné.

La distribution des classes dans le corpus, donnée en figure 3.7, s’établit à 16,3 % de Rouge, 25,8 % d’Orange, 28,5 % de Jaune, 22,2 % de Vert et 7,2 % de Bleu. Ce déséquilibre est voulu. Nous aurions pu équilibrer artificiellement les cinq classes à 20 % chacune ; le modèle aurait alors appris des probabilités a priori fausses, et notamment surestimé la fréquence des consultations non urgentes. La réalité d’un service d’urgences est celle que reflète notre distribution :

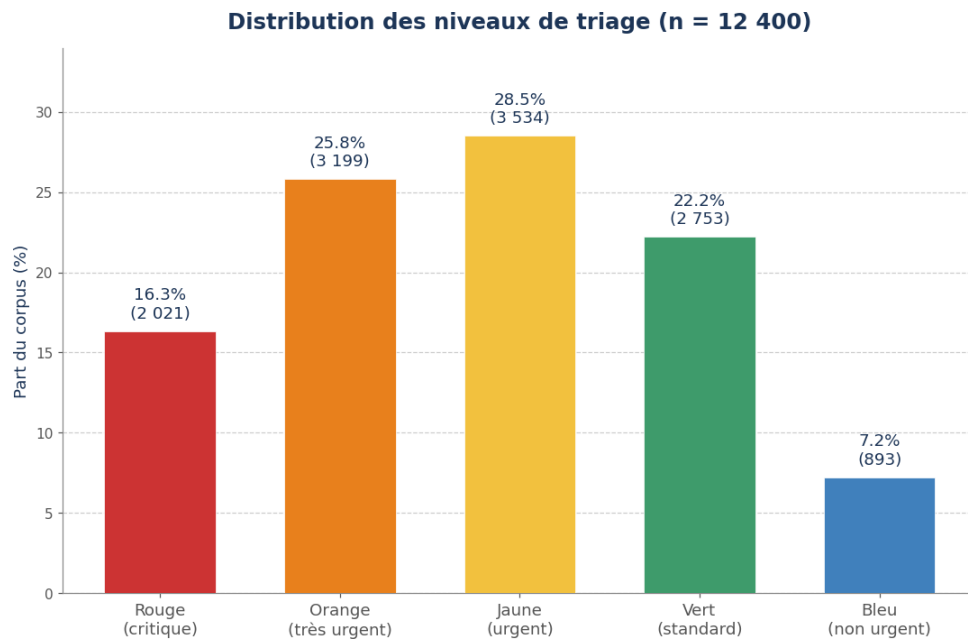


Fig. 3.7 : Distribution des classes de triage dans le corpus (n = 12 400).

les cas relevant d'un simple renouvellement d'ordonnance ou d'un suivi (Bleu) y sont minoritaires, et la masse des passages se concentre sur les niveaux intermédiaires. Nous avons veillé à conserver une représentation suffisante des classes extrêmes (Rouge et Bleu) pour que le modèle apprenne à les reconnaître malgré leur moindre fréquence.

3.5.4 Format des exemples d'entraînement

Reste à présenter ces cas au modèle dans le format qu'il comprend. Chaque exemple est sérialisé au format conversationnel de Mistral : l'instruction, encadrée par les balises [INST] et [/INST], contient le prompt système Manchester (décrit en 3.6.2) suivi de la fiche du patient ; la réponse attendue est le JSON de triage, et rien d'autre. Une première version du format de sortie imbriquait le résultat dans un objet intermédiaire ; nous l'avons aplatie en un format v2 dont les champs correspondent exactement à ceux de l'API du backend, ce qui supprime toute couche de transformation entre la sortie du modèle et la base de données. Voici un exemple abrégé, sur le cas classique de la douleur thoracique :

```
[INST] {prompt système Manchester}
Patient : 45 ans, Homme
Motif : Douleur thoracique
Symptômes : oppression, sueurs froides
Constantes : T=37,2 °C, TA=150/90, FC=110, SpO2=94 %, FR=22 [/INST]
{"level": 2, "color": "ORANGE", "confidence": 85,
 "diagnoses": [{"code": "I21.9",
  "name": "Infarctus du myocarde aigu", "probability": 70}],
 "recommendedTests": ["ECG", "Troponine"],
 "precautions": ["Monitoring continu", "Voie veineuse"],
```

```
"explanation": "Douleur thoracique d'allure coronarienne...", ...}
```

Le modèle apprend ainsi simultanément trois choses : la décision de triage elle-même, le respect strict d'un schéma de sortie exploitable par une machine, et la justification clinique en français qui sera montrée au médecin.

3.6 Notre modèle : BetterEMS-Triage

3.6.1 Choix du modèle de base

Le cahier des charges du modèle de fondation découle directement des exigences de 3.2 : il doit posséder des connaissances médicales solides, manier correctement le français, être disponible en poids ouverts pour un déploiement local, et tenir, entraînement compris, sur un unique GPU. Peu de candidats survivent à ces quatre filtres réunis.

Nous avons retenu BioMistral-7B [23], un modèle de 7,24 milliards de paramètres dérivé de Mistral-7B [22] par poursuite du pré-entraînement sur les publications biomédicales de PubMed Central. Ce pré-entraînement de domaine lui donne une familiarité réelle avec le vocabulaire et le raisonnement cliniques ; son ascendance Mistral lui assure une bonne tenue du français, vérifiée sur nos propres cas ; ses poids sont publics, et sa taille en fait l'un des rares modèles médicaux exécutables, et adaptables, sur un seul GPU comme celui du CRI. L'arithmétique mémoire est parlante : en demi-précision, le serveur disposait d'assez de mémoire pour charger le modèle en pleine précision tout en laissant la marge nécessaire aux activations et aux gradients des adaptateurs pendant l'entraînement. Les alternatives se sont écartées d'elles-mêmes. Mistral-7B généraliste exigeait de lui enseigner la médecine en plus du triage, tâche hors de portée d'un corpus de quelques milliers de cas. Med-PaLM, référence des modèles médicaux, est propriétaire et inaccessible hors du cloud de son éditeur [36], ce qui est incompatible avec notre contrainte de localité. Les grands modèles ouverts de 70 milliards de paramètres [21] débordent largement les capacités d'un GPU unique pour le fine-tuning. Les encodeurs de type BERT, enfin, auraient su classer mais pas générer la sortie riche que nous attendons, comme discuté au chapitre 2.

De cette base, nous avons fait notre modèle BetterEMS-Triage : le corpus d'adaptation, le prompt système, la procédure d'entraînement, l'export et le déploiement sont entièrement de notre conception. La filiation BioMistral demeure, et nous ne prétendons pas avoir pré-entraîné un modèle de zéro, mais le comportement final, lui, n'existe dans aucun modèle public : un moteur de triage Manchester qui raisonne en français sur des présentations cliniques algériennes et répond dans un format directement exploitable par notre backend.

3.6.2 Prompt système et sortie structurée

Le prompt système est la pièce qui transforme un modèle de langage en moteur de triage discipliné. Le nôtre assigne au modèle son rôle, assistant de triage suivant strictement le protocole

de Manchester, puis énumère, pour chacun des cinq niveaux, les discriminants cliniques qui y conduisent : voies aériennes compromises, état de choc ou crise convulsive en cours pour le Rouge; douleur sévère, douleur thoracique d'allure cardiaque, déficit neurologique aigu ou saturation inférieure à 95 % pour l'Orange ; et ainsi de suite jusqu'au Bleu des demandes administratives. La règle d'application est celle du protocole réel : le premier discriminant satisfait fixe le niveau [6].

La sortie est contrainte par un schéma JSON imposé, identique à celui des exemples d'entraînement : le niveau (level) et sa couleur (color), la confiance du modèle (confidence), le score de priorité et le délai d'attente maximal associés au niveau, les hypothèses diagnostiques (diagnoses) codées en CIM-10 avec leur probabilité, les examens recommandés (recommendedTests), les précautions de prise en charge (precautions), l'explication du raisonnement en français (explanation) et des drapeaux d'alerte (flags) : attention immédiate, spécialiste requis, isolement infectieux. Le prompt interdit explicitement tout texte hors du JSON.

Des règles de sécurité sont inscrites dans le prompt lui-même, avant même les garde-fous applicatifs de 3.7.2 : en cas de doute, toujours trier vers le niveau le plus urgent ; lorsque la confiance est basse, escalader d'un niveau et lever un drapeau de revue manuelle ; lorsque des signes de gravité contredisent une impression bénigne, suivre les signes. Le même prompt sert mot pour mot à l'entraînement et en production, car toute divergence entre les deux aurait placé le modèle, en service, devant des consignes qu'il n'a jamais vues.

3.6.3 Stratégie de fine-tuning

L'adaptation repose sur LoRA [24], dont le principe a été exposé au chapitre 2 : les 7,24 milliards de paramètres du modèle de base restent gelés, et l'apprentissage ne porte que sur de petites matrices de rang faible insérées en parallèle des poids existants. Nous avons placé ces adaptateurs sur l'ensemble des projections d'attention (q_proj, k_proj, v_proj, o_proj) et sur les couches feed-forward (gate_proj, up_proj, down_proj), une couverture plus large que la configuration minimale, qui ne touche que l'attention. Le total des paramètres entraînés atteint environ 84 millions, soit à peu près 1,1 % du modèle : c'est ce qui rend l'opération possible sur le serveur GPU du CRI.

La configuration retenue, résumée au tableau 3.3, n'est pas sortie d'un premier jet. Elle résulte d'une démarche expérimentale qui a confronté deux stratégies d'entraînement, QLoRA en quantification 4 bits [25] puis LoRA en pleine précision fp16, dont le déroulement, les courbes et le diagnostic sont analysés en détail au chapitre 4. Nous nous bornons ici à la configuration gagnante.

Tab. 3.3 : Hyperparamètres de fine-tuning de BetterEMS-Triage (configuration retenue).

Hyperparamètre	Valeur retenue
Rang LoRA r	32
Facteur d'échelle α	64
Dropout des adaptateurs	0,05
Taux d'apprentissage	2e-5, décroissance cosinus, 5 % de warmup
Batch	1, accumulation de gradient 16 (batch effectif 16)
Époques	3
Optimiseur	AdamW
Écrêtage du gradient	norme maximale 1,0
Précision de calcul	fp16
Modules adaptés	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj

Quelques choix demandent un mot. Le batch effectif de 16 est obtenu par accumulation de gradient : la mémoire du serveur GPU n'autorise qu'un exemple à la fois en fp16, mais les gradients de seize exemples sont accumulés avant chaque mise à jour, ce qui restitue la stabilité d'un batch plus large. Le taux d'apprentissage de 2e-5, prudent, et l'écrêtage du gradient à 1,0 répondent à la taille modeste du corpus au regard du modèle : sur 12 400 exemples, un pas d'apprentissage trop franc mémorise au lieu de généraliser. La validation est évaluée à chaque époque et le meilleur point de contrôle est conservé automatiquement, une précaution dont le chapitre 4 montrera qu'elle a été décisive. L'implémentation s'appuie sur PyTorch [38], la bibliothèque Transformers [39] et PEFT [40] ; l'entraînement complet s'est déroulé sur le serveur GPU du CRI, sur trois passes complètes du corpus.

3.6.4 Export et quantification

Un modèle entraîné dans un environnement de recherche n'est pas encore un modèle de production. La chaîne d'export comporte trois étapes. Les adaptateurs LoRA sont d'abord fusionnés dans les poids du modèle de base (`merge_and_unload`) : les matrices de rang faible sont additionnées aux poids gelés, et le surcoût d'inférence de LoRA disparaît ; le résultat est un modèle ordinaire, indiscernable d'un modèle entraîné classiquement. Ce modèle fusionné est ensuite converti au format GGUF en pleine précision F16 [37]. Il subit enfin une quantification Q4_K_M qui le compresse sensiblement pour le déploiement.

Le choix de Q4_K_M parmi les schémas de quantification disponibles tient à son compromis : il code la majorité des poids sur quatre bits mais préserve une précision supérieure pour les couches les plus sensibles, ce qui limite la dégradation à un niveau imperceptible à l'usage [26],

et le chapitre 4 le vérifie sur nos cas de test. Pour la production, les bénéfices sont directs : un fichier unique et autonome, un chargement rapide, et une empreinte mémoire qui laisse à la carte de quoi servir les requêtes confortablement. C'est ce fichier quantifié, et lui seul, qui constitue l'artefact déployé : tout l'environnement d'entraînement Python reste au laboratoire.

3.7 Intégration de l'IA dans le système

3.7.1 Service d'inférence

En production, le modèle est servi par llama-server, le serveur d'inférence de l'écosystème llama.cpp compilé avec le support CUDA [37]. Il expose une API compatible OpenAI (/v1/chat/completions), ce qui présente un avantage pratique considérable : le backend lui parle avec un client HTTP standard, et le moteur pourrait être remplacé par n'importe quel service respectant la même interface sans toucher au code métier. En service, le modèle n'occupe qu'une fraction de la mémoire vidéo du serveur, contexte de génération compris.

Côté backend, le module ai orchestre l'appel. Il assemble la fiche du patient à partir des données de la visite, exactement dans le gabarit vu à l'entraînement (3.5.4), l'envoie au serveur d'inférence à travers le VPN, puis traite la réponse. Le JSON retourné n'est jamais pris pour argent comptant : sa structure est analysée champ par champ, les champs manquants ou hors domaine sont normalisés (couleur déduite du niveau, confiance bornée à 0-100, délai dérivé du niveau), et un JSON entièrement illisible est rabattu sur une proposition ORANGE de sécurité. Si le serveur d'inférence est injoignable, le module bascule sur le mécanisme de repli décrit ci-dessous. Le résultat validé est enregistré comme TriageResult et le patient rejoint la file. L'ensemble tient en quelques secondes par patient ; les mesures précises de débit et de latence sont présentées au chapitre 4.

3.7.2 Mécanismes de sécurité

Mettre un modèle de langage dans la chaîne de décision d'un service d'urgences impose une discipline que nous avons traitée comme un objet de conception à part entière, et non comme une couche ajoutée après coup. La sécurité de BetterEMS est construite en profondeur, sur cinq niveaux qui se recouvrent mutuellement : la défaillance de l'un est rattrapée par les autres.

Le premier niveau est le principe directeur du système : l'IA propose, le médecin dispose. Aucun patient n'est orienté, traité ou renvoyé sur la seule sortie du modèle ; chaque triage attend la confirmation ou la correction d'un médecin, et l'interface est conçue pour que cette validation soit un geste rapide, pas une corvée contournable. Cette place centrale de l'humain rejoint les recommandations de l'OMS sur l'IA en santé [30] et la littérature sur le déploiement responsable de l'apprentissage automatique en clinique [31]. Le deuxième niveau est le seuil de confiance : la consigne du prompt demande au modèle de signaler toute confiance inférieure à 60, et un

garde-fou applicatif plafonne l'urgence à ORANGE et marque le cas pour revue manuelle dès que la confiance passe sous le seuil configuré (50), car le doute profite toujours au patient. Le troisième est le repli algorithmique : si le moteur d'inférence est indisponible, trop lent, ou si sa réponse échoue à la validation, un algorithme déterministe à 8 règles par mots-clés prend le relais. Il est volontairement fruste et conservateur, et il sur-trie, mais il garantit que le service de triage ne s'arrête jamais. Le quatrième niveau est le principe de précaution lui-même, encodé à tous les étages : dans les cas d'entraînement, dans les règles du prompt, dans le repli. L'asymétrie est assumée : un sur-triage coûte un peu d'attente aux autres patients ; un sous-triage peut coûter une vie. Le cinquième, enfin, est la traçabilité : le module d'audit journalise chaque action (qui a enregistré, qui a saisi les constantes, ce que l'IA a proposé et avec quelle confiance, ce que le médecin a décidé) avec l'utilisateur, son rôle et l'horodatage. N'importe quel triage peut être rejoué a posteriori, ce qui est la condition de toute analyse d'incident comme de toute démarche qualité.

Pris isolément, aucun de ces mécanismes n'est spectaculaire. C'est leur empilement qui fait, selon nous, l'un des points forts du système : BetterEMS reste utilisable, sûr et auditable y compris le jour où son composant le plus sophistiqué tombe en panne.

3.7.3 File d'attente temps réel

La file d'attente matérialise ce que le chapitre 1 décrivait en théorie : une file à priorité, où le niveau de Manchester prime et où l'ancienneté départage. Concrètement, chaque QueueEntry porte le niveau de triage et l'heure d'entrée dans la file ; le tri s'effectue d'abord par niveau, puis, à niveau égal, par ordre d'arrivée. Un patient Orange enregistré à l'instant passe devant tous les Jaunes, mais derrière les Orange qui attendaient déjà : la règle est simple, et surtout elle est visible et explicable au personnel.

Le temps réel repose sur la passerelle Socket.io du module queue. À chaque événement (nouveau triage, appel d'un patient, fin de consultation, correction d'un niveau), le serveur pousse l'état mis à jour vers tous les clients connectés : les postes des médecins voient la file évoluer sans rechargement, et l'écran TV de la salle d'attente réagit dans la seconde. Cet écran affiche les patients appelés avec leur numéro de ticket et le box où se présenter, accompagnés d'un signal sonore à chaque appel. L'affichage ne montre que le ticket, jamais l'identité : la salle d'attente est un lieu public, et le numéro remis à l'enregistrement suffit au patient pour se reconnaître.

Le délai cible attaché à chaque niveau, immédiat pour le Rouge, dix minutes pour l'Orange, jusqu'à 240 minutes pour le Bleu, accompagne chaque entrée de file, si bien que le système connaît à tout instant l'écart entre l'attente réelle d'un patient et l'objectif du protocole [6]. Les délais théoriques du chapitre 1 deviennent ainsi des grandeurs observables, patient par patient, dans le tableau de bord du service.

3.7.4 Gestion des rôles et des accès

Le contrôle d'accès traduit en code la séparation des métiers décrite en 3.2. L'authentification repose sur des jetons JWT délivrés à la connexion; chaque jeton embarque le rôle de l'utilisateur, et des gardes placées sur chaque point d'entrée de l'API appliquent un contrôle d'accès par rôle (RBAC). La règle est celle du moindre privilège : l'agent d'accueil crée des patients et des visites mais ne touche pas aux données cliniques ; l'infirmier saisit constantes et symptômes et déclenche le triage ; le médecin est le seul à pouvoir appeler un patient, valider ou corriger un triage et clore une visite ; l'ambulancier du SAMU n'accède qu'au pré-enregistrement; l'administrateur gère les comptes, les paramètres et consulte les statistiques, sans intervenir dans le flux de soins.

Le frontend applique la même partition : les onze pages de l'application sont filtrées par rôle, et un utilisateur ne voit même pas les écrans qui ne le concernent pas. Cette double application, côté serveur, qui fait autorité, et côté interface, qui guide, se combine avec le module d'audit : puisque chaque action est journalisée avec son auteur et son rôle, la question « qui a fait quoi » a toujours une réponse.

3.8 Apprentissage continu

Un système de triage figé au jour de son déploiement vieillirait mal : les pratiques évoluent, les pathologies saisonnières tournent, et les faiblesses du modèle ne se révèlent qu'à l'usage. Nous avons donc conçu BetterEMS pour que son fonctionnement normal produise, sans effort supplémentaire, la matière de sa propre amélioration.

Le mécanisme tient à la validation médicale elle-même. Chaque fois qu'un médecin confirme ou corrige un triage (niveau retenu, diagnostics validés ou écartés), sa décision est enregistrée dans l'entité DoctorFeedback, en regard de la proposition du modèle. Il n'y a là aucune saisie additionnelle : le geste de validation, obligatoire de toute façon, est le geste de collecte. De cette confrontation systématique entre la machine et le praticien découle un indicateur que le tableau de bord expose : le taux d'accord médecin/IA, globalement et par niveau. C'est, à nos yeux, la mesure de qualité qui compte vraiment en exploitation, bien plus parlante pour un chef de service qu'une métrique d'apprentissage automatique.

Ces retours alimentent une boucle périodique : les cas corrigés, c'est-à-dire ceux où le médecin a contredit le modèle, sont les exemples les plus instructifs qui soient, puisque, par construction, ce sont les erreurs réelles du système. Convertis au format d'entraînement de 3.5.4, ils viennent enrichir le corpus pour un nouveau cycle de fine-tuning, suivi d'une évaluation sur l'ensemble de test gelé, puis d'un redéploiement si, et seulement si, la nouvelle version fait mieux que l'ancienne. Le versant collecte de cette boucle est conçu et implémenté : les retours s'accumulent en base, structurés et datés, prêts à l'emploi. L'automatisation du ré-entraînement périodique constitue l'étape suivante de la feuille de route ; elle réutilisera tels quels les scripts d'entraînement et d'export du présent chapitre.

3.9 Conclusion

Ce chapitre a parcouru l'ensemble de ce que nous avons construit. Un système hospitalier complet d'abord : cinq rôles, neuf entités, onze interfaces, une file d'attente temps réel et un écran de salle d'attente, déployés sur une architecture à deux serveurs où les données de santé restent sous le contrôle de l'établissement. Un corpus propriétaire ensuite : 12 400 cas de triage rédigés en français à la main, nettoyés et découpés pour l'entraînement de notre modèle, dans le respect de la distribution réelle d'un service d'urgences. Notre modèle BetterEMS-Triage enfin : 7,24 milliards de paramètres adaptés par LoRA sur ce corpus, quantifiés pour la production, et intégrés derrière cinq niveaux de garde-fous où le médecin conserve, toujours, le dernier mot.

Chaque choix présenté ici, qu'il s'agisse de la configuration d'entraînement, du format de sortie, de la quantification ou du repli algorithmique, engage une promesse de performance qu'il reste à honorer chiffres en main. C'est l'objet du chapitre 4 : la confrontation des deux stratégies de fine-tuning et le diagnostic qui a départagé l'une de l'autre, les courbes d'apprentissage et la sélection du meilleur point de contrôle, la précision mesurée en validation, et la vitesse d'inférence qui décide, en dernière analyse, de la viabilité du système au comptoir des urgences.

Expérimentations et résultats

Le chapitre précédent a présenté l’architecture de BetterEMS et la constitution de notre corpus propriétaire de 12 400 cas cliniques. Il reste à démontrer que l’ensemble fonctionne, et à quel niveau de performance. C’est l’objet de ce chapitre. Nous y décrivons d’abord l’environnement expérimental mis à notre disposition par le CRI de l’Université de Saïda, puis la démarche qui nous a conduits de la première stratégie de fine-tuning testée à la configuration finalement retenue. Nous détaillons ensuite l’entraînement de notre modèle BetterEMS-Triage, ses courbes d’apprentissage et les chiffres qui en résultent, avant de mesurer ses performances d’inférence en conditions de production, de le confronter à l’algorithme à règles qui lui sert de solution de repli, et de valider le système complet sur un scénario de bout en bout. Un bilan ferme le chapitre et prépare la transition vers l’étude entrepreneuriale.

4.1 Environnement expérimental

L’adaptation d’un modèle de 7,24 milliards de paramètres ne se fait pas sur un poste de travail ordinaire. L’ensemble des entraînements a été conduit sur le serveur GPU du CRI (Centre de Recherche en Informatique) de l’Université de Saïda, conçu pour soutenir des charges de calcul ininterrompues de plusieurs heures, condition nécessaire pour un entraînement qui, on le verra, a duré plusieurs heures d’une traite. Sa mémoire vidéo s’est par ailleurs révélée déterminante dans le choix de notre stratégie d’entraînement, comme la section 4.2 le montrera.

Côté logiciel, la pile d’entraînement repose sur Python et PyTorch [38], complétés par la bibliothèque Transformers [39] pour le chargement du modèle et la boucle d’entraînement, et par PEFT [40] pour l’injection des adaptateurs LoRA. L’inférence en production s’appuie sur un outillage différent, optimisé pour la latence : llama.cpp et son serveur llama-server [37], compilés avec le support CUDA, exposent le modèle derrière une API compatible OpenAI que le backend interroge directement.

Les données proviennent du pipeline décrit au chapitre 3. L’ensemble d’entraînement compte 12 400 cas rédigés à la main ; les ensembles de validation et de test comptent 100 cas chacun, tenus à l’écart de l’entraînement afin qu’aucun chevauchement ne fausse l’évaluation. Chaque cas (avec son prompt système, sa fiche patient et sa sortie JSON attendue) est parcouru à chacune des trois passes sur le corpus. Le tableau 4.1 récapitule cet environnement.

Tab. 4.1 : Environnement matériel et logiciel de la campagne expérimentale.

Composant	Détail
GPU	Serveur GPU du CRI de l’Université de Saïda
Pile d’entraînement	Python, PyTorch [38], Transformers [39], PEFT [40]
Pile d’inférence	llama.cpp / llama-server [37], compilé CUDA, API compatible OpenAI
Corpus d’entraînement	12 400 cas cliniques
Validation / test	100 cas / 100 cas, isolés de l’entraînement

Cet environnement appelle une remarque. Un seul serveur GPU reste une ressource modeste à l’échelle des entraînements de modèles de langage : la plupart des travaux publiés mobilisent des grappes entières. Toute notre démarche expérimentale a donc été pensée pour tirer le maximum de ce serveur unique. C’est précisément cette contrainte qui a structuré la comparaison des deux stratégies de fine-tuning présentée ci-dessous.

4.2 Démarche expérimentale : deux stratégies de fine-tuning comparées

Adapter un modèle de 7,24 milliards de paramètres sur un seul GPU pose une question d’arbitrage classique : faut-il sacrifier de la précision numérique pour gagner de la mémoire, ou occuper davantage de mémoire pour préserver la stabilité de l’entraînement ? Nous avons traité cette question comme une véritable expérience, avec une hypothèse de départ, des mesures, un diagnostic et une décision.

Notre première hypothèse privilégiait l’économie de mémoire. QLoRA [25] combine les adaptateurs basse-rang de LoRA avec une quantification du modèle de base en 4 bits au format NF4 : le modèle gelé occupe alors nettement moins de mémoire, ce qui laisse une marge confortable sur n’importe quel GPU. C’est la voie standard recommandée lorsque la mémoire est comptée, et l’argument économique était réel : avec un modèle de base ainsi allégé, l’entraînement aurait tenu sur une machine bien plus modeste que la nôtre. Nous l’avons donc testée en premier, avec un rank LoRA de 64 et un taux d’apprentissage de $2e-4$, valeurs usuelles pour cette méthode.

Le résultat a été net : au lieu de décroître, la perte d’entraînement a divergé dès l’époque 0,2, soit un cinquième de la première passe sur les données. Plutôt que de conclure hâtivement, nous avons cherché à isoler la cause. Le suspect habituel d’une divergence est un taux d’apprentissage trop élevé ; nous avons donc relancé l’entraînement avec un taux divisé par quatre ($5e-5$), puis encore réduit à $2e-5$. La divergence s’est reproduite à l’identique dans les deux cas, au même point de l’entraînement. Le taux d’apprentissage était donc hors de cause.

Ce protocole d’élimination nous a conduits au diagnostic suivant : l’instabilité provient de la quantification 4 bits elle-même. Les erreurs d’arrondi introduites par la représentation NF4 du modèle gelé se propagent dans les gradients à chaque mise à jour ; sur un corpus généraliste massif, ce bruit se moyenne et reste contenu, mais sur un corpus spécialisé de quelques milliers d’exemples, où chaque batch tire fortement les poids dans la même direction, il s’accumule jusqu’à faire diverger la perte [25]. Le tableau 4.2 résume la configuration testée et son issue.

Tab. 4.2 : Configuration QLoRA testée (hypothèse 1) et issue des trois essais.

Paramètre	Valeur
Méthode	QLoRA, quantification NF4 4 bits
Rank LoRA	64
Taux d’apprentissage testés	2e-4, puis 5e-5, puis 2e-5
Corpus	cas de triage, format initial
Résultat	divergence de la perte dès l’époque 0,2, reproduite aux trois taux

Restait à en tirer la conséquence. Puisque le problème venait de la quantification, la solution consistait à la supprimer : charger le modèle de base en pleine précision fp16 et n’entraîner que les adaptateurs LoRA par-dessus. Cette option est inaccessible sur les GPU grand public ; elle devenait possible sur le serveur GPU du CRI, qui disposait d’assez de mémoire pour charger le modèle en pleine précision en laissant la marge nécessaire aux activations et aux états de l’optimiseur. Nous avons donc basculé sur cette configuration, qui a convergé du premier coup. C’est elle qui a produit BetterEMS-Triage, détaillée dans la section suivante.

Cette campagne nous a livré un enseignement méthodologique qui dépasse notre cas d’usage : sur un corpus spécialisé de taille réduite, la robustesse numérique du fp16 l’emporte sur l’économie de mémoire du 4 bits, et le diagnostic ne peut être posé qu’en éliminant systématiquement les causes alternatives, ici trois taux d’apprentissage testés avant d’incriminer la quantification. À notre connaissance, ce point est rarement documenté dans la littérature appliquée, où QLoRA est souvent présenté comme un remplacement sans coût de LoRA.

4.3 Entraînement de BetterEMS-Triage (LoRA fp16)

4.3.1 Configuration retenue

Notre modèle BetterEMS-Triage est construit par adaptation LoRA [24] de BioMistral-7B [23], le modèle de base étant chargé en fp16 et entièrement gelé. Les adaptateurs sont insérés dans les quatre projections d’attention (q_proj, k_proj, v_proj, o_proj) et dans les trois couches du bloc feedforward (gate_proj, up_proj, down_proj) de chaque couche du Transformer, soit environ

84 millions de paramètres entraînaibles, 1,1 % des 7,24 milliards de paramètres du modèle. Le tableau 4.3 rassemble les hyperparamètres retenus.

Tab. 4.3 : Hyperparamètres de l'entraînement de BetterEMS-Triage.

Hyperparamètre	Valeur
Précision du modèle de base	fp16 (sans quantification)
Rank / alpha LoRA	32 / 64
Dropout LoRA	0,05
Taux d'apprentissage	2e-5, ordonnancement cosinus, 5 % de warmup
Batch	1, accumulation de gradient 16 (batch effectif 16)
Époques	3
Optimiseur	AdamW, gradient clipping 1,0
Modules adaptés	q/k/v/o_proj + gate/up/down_proj ($\approx$ 84 M paramètres, 1,1 %)

Trois choix méritent un mot. Par rapport à l'essai QLoRA, nous avons ramené le rank des adaptateurs de 64 à 32 : cette capacité reste largement suffisante pour une tâche de classification structurée, et des matrices plus petites réduisent d'autant la tendance à mémoriser le corpus, une précaution dont la section suivante montrera qu'elle était justifiée. Le batch effectif de 16, ensuite, est obtenu par accumulation de gradient : la carte ne peut traiter qu'un exemple à la fois en fp16, mais les gradients de seize exemples successifs sont sommés avant chaque mise à jour, ce qui restitue la stabilité statistique d'un vrai batch. Le taux d'apprentissage de 2e-5, enfin, dix fois plus bas que la valeur usuelle de QLoRA, prolonge la prudence acquise lors de la première campagne : avec un corpus spécialisé, mieux vaut des pas courts et un ordonnancement cosinus qui les réduit encore en fin d'entraînement.

L'entraînement complet a duré plusieurs heures, à raison de trois passes sur le corpus. La mémoire du serveur a absorbé sans incident les variations de longueur des séquences; aucun entraînement n'a été interrompu pour dépassement mémoire.

4.3.2 Courbes d'apprentissage

Les figures 4.1 et 4.2 présentent les courbes obtenues, tracées à partir des journaux d'entraînement réels.

Le tableau 4.4 donne les valeurs mesurées en fin de chaque époque.

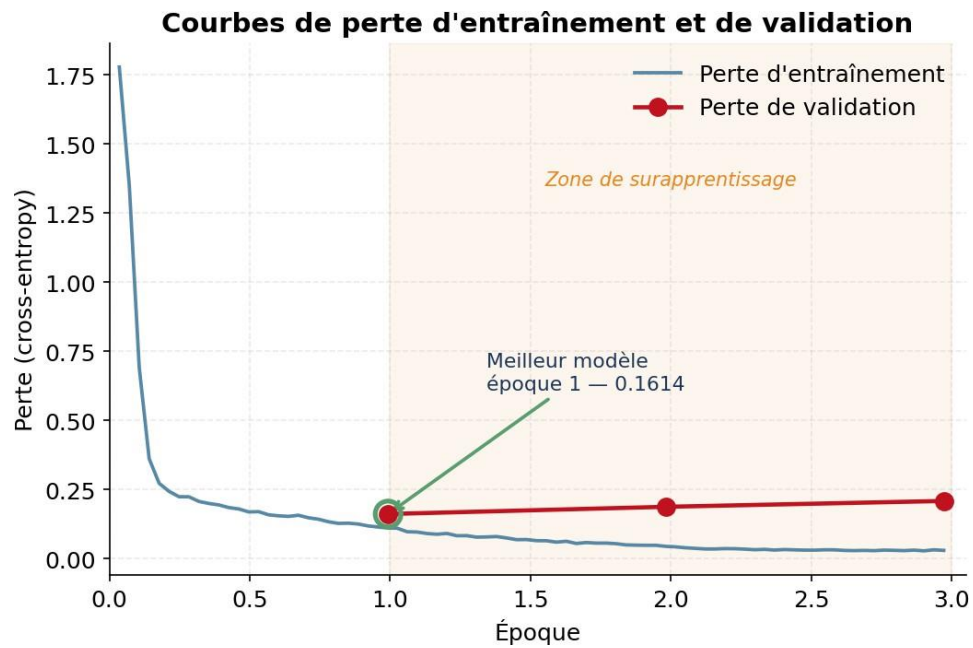


Fig. 4.1 : Courbes de perte d'entraînement et de validation (3 époques, données réelles).

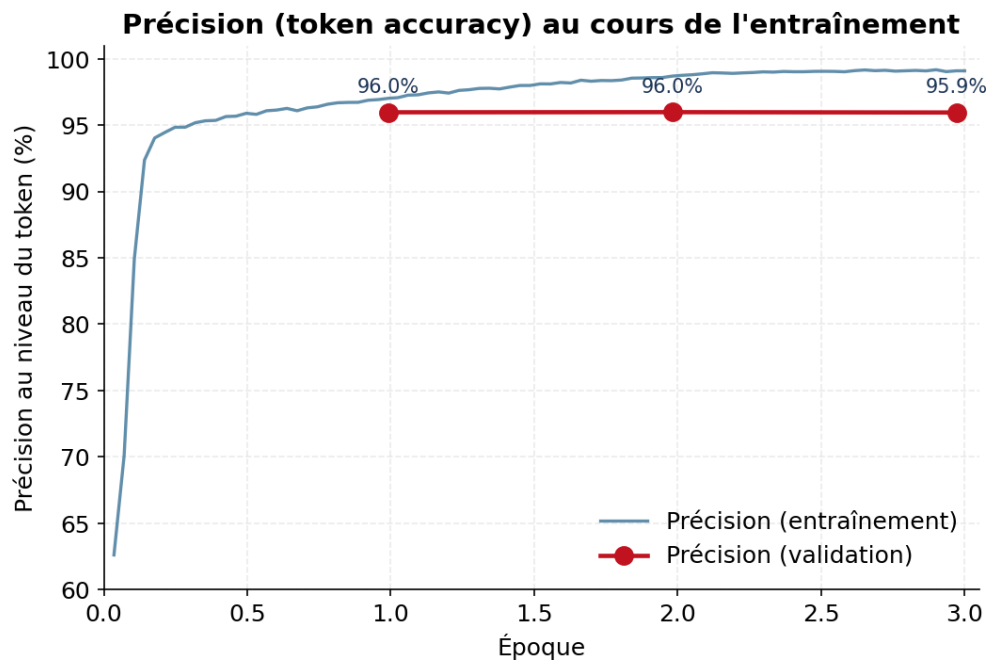


Fig. 4.2 : Token accuracy au cours de l'entraînement (données réelles).

Tab. 4.4 : Pertes et précision par époque; le meilleur point de validation est atteint à l'époque 1.

Métrique	Époque 1	Époque 2	Époque 3
Perte d'entraînement	0,2671	0,0476	0,0144
Perte de validation	0,1614	0,1874	0,2085
Token accuracy (validation)	0,9596	0,9597	0,9595

La lecture de ces courbes raconte un entraînement qui s'est déroulé exactement comme la théorie du chapitre 2 le laissait prévoir. Pendant la première époque, les deux pertes baissent de concert : le modèle apprend la structure de la tâche (le format JSON de sortie, la logique des cinq niveaux de Manchester, l'association entre constantes vitales et gravité), et cette connaissance se transfère intégralement aux cas de validation qu'il n'a jamais vus. La perte de validation atteint son minimum de 0,1614 en fin d'époque 1.

À partir de la deuxième époque, les trajectoires se séparent. La perte d'entraînement continue de chuter, de 0,2671 à 0,0476 puis 0,0144 : le modèle mémorise de mieux en mieux ses 12 400 exemples. Mais la perte de validation remonte, à 0,1874 puis 0,2085 : ce qu'il gagne en mémorisation ne se généralise plus. C'est le surapprentissage classique d'un corpus spécialisé de taille moyenne, phénomène décrit au chapitre 2 sous l'angle du surapprentissage et de la capacité du modèle. Sur un corpus de quelques milliers d'exemples, un modèle de 7,24 milliards de paramètres, même bridé à 84 millions de paramètres ajustables, dispose d'une capacité largement suffisante pour apprendre chaque cas par cœur ; la question n'est pas d'éviter le surapprentissage, mais de s'arrêter au bon moment.

Notre protocole avait anticipé ce scénario. L'option `load_best_model_at_end=True` de la boucle d'entraînement évalue le modèle à chaque fin d'époque et recharge automatiquement, à l'issue des trois époques, le checkpoint dont la perte de validation est minimale, ici celui de l'époque 1, à 0,1614. Le modèle livré n'est donc pas le dernier état de l'entraînement mais le meilleur, sélectionné sur un critère objectif et sans intervention manuelle. Ce mécanisme est la traduction opérationnelle de l'arrêt précoce présenté au chapitre 2 parmi les techniques de régularisation : plutôt que de brider la capacité du modèle a priori, on le laisse apprendre librement et on retient l'état où sa généralisation était maximale. Les époques 2 et 3 n'ont d'ailleurs pas été du temps perdu : elles confirment que le minimum de l'époque 1 est bien un minimum, et non un simple point de passage sur une courbe qui aurait pu redescendre.

La courbe de précision complète ce tableau. Dès la première époque, le modèle atteint ≈ 96 % de précision (token accuracy) sur l'ensemble de validation, et y reste remarquablement stable : 0,9596, 0,9597, 0,9595 sur les trois époques. Cette stabilité, alors même que la perte de validation remonte, s'explique : la perte mesure la confiance du modèle sur chaque token (et se dégrade quand il devient trop sûr de lui sur des cas mémorisés), tandis que la précision mesure le choix du token le plus probable, qui reste correct. Concrètement, le modèle produit un JSON dont environ 96 % des tokens correspondent exactement à la sortie attendue : structure, codes

CIM-10, niveaux et délais compris.

4.3.3 Portée de la mesure

Cette valeur de $\approx 96\%$ s'interprète pour ce qu'elle est : une précision au niveau du token, mesurée sur la sortie JSON complète de l'ensemble de validation, qui atteste que le modèle a appris le format et la logique du triage. L'évaluation détaillée par niveau de gravité (matrice de confusion 5×5 et score F1 par classe sur les 100 cas de test) constitue la prochaine étape de notre feuille de route d'évaluation, et son protocole est déjà défini.

4.4 Performances d'inférence

Un modèle de triage n'a de valeur que s'il répond dans le temps d'un passage à l'infirmier. Une fois l'entraînement terminé, nous avons fusionné les adaptateurs LoRA dans les poids de base (merge_and_unload), exporté le modèle au format GGUF, puis quantifié le résultat en Q4_K_M [26] pour le déploiement, sans dégradation perceptible des sorties sur les cas de test : les niveaux assignés et les diagnostics proposés restent identiques à ceux du modèle pleine précision. Le format Q4_K_M n'applique d'ailleurs pas une quantification uniforme : les couches d'attention, les plus sensibles, conservent une précision supérieure à celle des couches feedforward, ce qui explique en partie que la quantification préserve aussi bien la qualité des décisions. On notera que cette quantification d'inférence ne contredit pas le diagnostic de la section 4.2 : les erreurs d'arrondi qui s'accumulaient dans les gradients pendant l'entraînement sont sans effet lorsqu'on se contente de lire les poids.

Servi par llama-server [37] sur le serveur GPU du CRI, le modèle est absorbé rapidement à la lecture du prompt et génère ensuite le JSON de sortie. Le prompt d'un cas de triage (prompt système, fiche patient, constantes, symptômes) est ainsi traité en moins d'une seconde, et la génération du JSON complet prend le reste : au total, deux à quatre secondes par patient selon la longueur de la réponse.

Le prompt d'un cas, qui réunit le prompt système, la fiche patient, les constantes et les symptômes, représente quelques centaines de tokens absorbés en une fraction de seconde ; la génération du JSON de sortie occupe l'essentiel du temps de réponse.

Ces chiffres prennent tout leur sens rapportés au flux réel d'un service. Le triage est demandé au moment où l'infirmier valide la saisie des constantes ; une réponse en deux à quatre secondes arrive avant même que le patient ait quitté le poste de prise des constantes, et s'efface dans le geste de saisie lui-même. C'est ce qui sépare un outil adopté d'un outil contourné : une assistance qui suit le rythme du soignant plutôt que de le lui imposer. En période d'affluence, un système qui ferait attendre finirait débranché.

La conclusion opérationnelle est nette : la contrainte temps réel d'un service d'urgences est tenue avec un seul GPU. La même machine qui a servi à l'entraînement héberge le service

d’inférence en gardant une large réserve de mémoire. Pour un établissement de santé, le coût matériel du déploiement se résume donc à un GPU unique, un ordre de grandeur remarquablement contenu pour un système d’aide à la décision fondé sur un modèle de 7,24 milliards de paramètres.

4.5 Comparaison avec l’algorithme à règles

BetterEMS embarque une solution de repli : un algorithme à 8 règles par mots-clés, décrit au chapitre 3, qui prend le relais si le modèle est indisponible ou si sa confiance passe sous le seuil configuré. Cette redondance est une exigence de sûreté ; elle offre aussi un point de comparaison instructif, puisque les deux approches traitent exactement les mêmes données d’entrée.

L’algorithme à règles a des qualités réelles. Il répond instantanément, son comportement est entièrement prévisible, et chaque décision se justifie par la règle qui l’a déclenchée. Mais sa logique est rigide par construction. Chaque règle teste la présence d’un mot-clé dans le motif ou les symptômes ; il ignore les combinaisons de constantes (une tension limite associée à une fréquence cardiaque limite ne déclenche rien si aucune des deux ne franchit son seuil propre), il est aveugle aux formulations libres qui ne contiennent pas le mot attendu, et les présentations atypiques passent entre les mailles de règles écrites pour les présentations typiques.

Notre modèle travaille différemment : il lit le motif de consultation et les symptômes comme du texte libre et les interprète dans le contexte des constantes. Les cas de test fournissent des illustrations parlantes. Un patient décrivant une oppression et une gêne irradiant vers le bras, avec des sueurs, sans que le mot « douleur thoracique » figure dans le motif, ne déclenche aucune des 8 règles et serait classé par défaut ; le modèle, lui, reconnaît dans cette formulation atypique un tableau coronarien possible et classe le cas ORANGE, avec un syndrome coronarien aigu en tête des diagnostics proposés et un ECG en premier examen recommandé. L’explication rédigée qui accompagne la décision donne à l’infirmier les éléments du raisonnement, là où l’algorithme à règles n’aurait rien eu à dire. C’est exactement le type de cas, grave mais mal formulé, pour lequel le triage assisté se justifie [6]. Notons que le comportement du modèle reste encadré par les règles de sûreté inscrites dans son prompt système : en cas d’incertitude, sur-trier plutôt que sous-trier, et signaler une confiance basse pour déclencher l’escalade vers le fallback. Le tableau 4.5 systématise cette comparaison, en restant volontairement qualitatif.

Tab. 4.5 : Comparaison qualitative entre l’algorithme à règles et notre modèle sur les cas de test.

Critère	Algorithme à 8 règles	BetterEMS-Triage
Temps de réponse	immédiat	2–4 s
Compréhension du motif	mots-clés exacts uniquement	texte libre, synonymes, formulations atypiques
Constantes vitales	seuils testés isolément	interprétées en combinaison et en contexte

Critère	Algorithme à 8 règles	BetterEMS-Triage
Cas atypiques	non couverts par construction	correctement traités sur les cas de test examinés
Sortie produite	niveau de triage seul	JSON complet : niveau, diagnostics CIM-10, examens, précautions, explication
Explicabilité	règle déclenchée identifiable	explication rédigée pour chaque décision
Prévisibilité	totale	élevée, bornée par le seuil de confiance

Sur les cas de test que nous avons examinés manuellement, le modèle identifie correctement les présentations RED et ORANGE, la propriété qui compte le plus en pratique puisque c'est là que se joue la sécurité du patient, les confusions résiduelles se concentrant aux frontières ORANGE/JAUNE et VERT/BLEU, où les protocoles humains eux-mêmes divergent souvent [7]. Cette lecture reste qualitative : sa confirmation chiffrée passe par la matrice de confusion 5×5 et les F1 par niveau annoncés en 4.3.3. L'architecture du système tire parti de cette complémentarité : le modèle assure la finesse d'interprétation, les règles garantissent qu'aucune panne ne laisse jamais le service sans triage.

4.5.1 Évaluation chiffrée de l'algorithme de repli

Au-delà de la comparaison qualitative, l'algorithme de repli se prête à une évaluation de classification complète, puisqu'il est déterministe et qu'il s'exécute sans GPU. Nous l'avons appliqué aux 100 cas de test, restés à l'écart de l'entraînement, et nous avons confronté ses prédictions aux niveaux de référence. Cette mesure établit la performance de base que le modèle de langage doit dépasser.

Les résultats confirment la rigidité pressentie. L'algorithme atteint une exactitude stricte de 38 % et classe 81 % des cas à au plus un niveau du bon ; surtout, son taux de sous-triage s'élève à 27 %, c'est-à-dire qu'un cas sur quatre se voit attribuer une urgence inférieure à la sienne, ce qui est rédhibitoire pour un outil de triage. La matrice de confusion de la figure 4.3 en donne la lecture détaillée : faute de mot-clé reconnu, de nombreux cas tombent dans la classe Jaune par défaut, et la classe Vert n'est jamais correctement identifiée. Le tableau 4.6 reporte la précision, le rappel et le F1 par niveau.

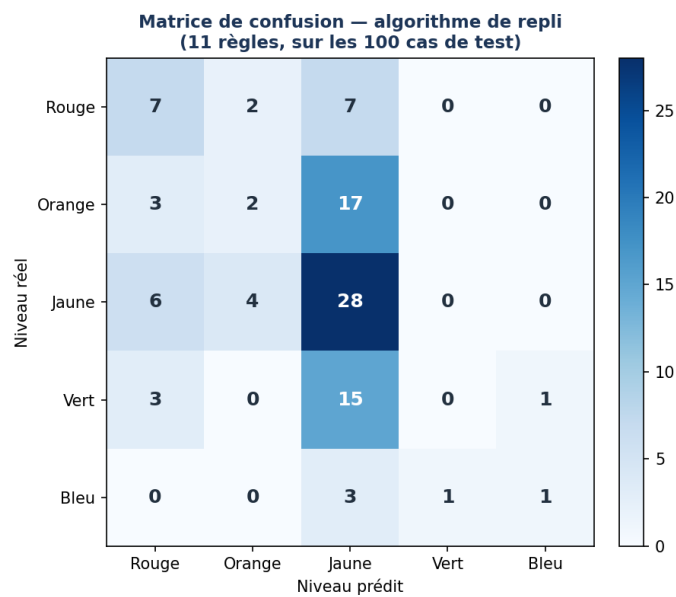


Fig. 4.3 : Matrice de confusion de l’algorithme de repli sur les 100 cas de test (lignes : niveau réel ; colonnes : niveau prédit).

Tab. 4.6 : Performance par niveau de l’algorithme de repli sur l’ensemble de test (n = 100).

Niveau	Précision	Rappel	F1
Rouge	0,37	0,44	0,40
Orange	0,25	0,09	0,13
Jaune	0,40	0,74	0,52
Vert	0,00	0,00	0,00
Bleu	0,50	0,20	0,29

Ces chiffres ne sont pas un échec : ils sont la raison d’être du modèle de langage. Un algorithme à mots-clés, aussi soigné soit-il, ne capture ni les combinaisons de constantes ni les formulations libres, et son taux de sous-triage le rend impropre à un usage autonome. Il remplit parfaitement son rôle de filet de sécurité, jamais celui de moteur principal.

Le même protocole, appliqué cette fois au modèle BetterEMS-Triage, fournira la matrice de confusion et les F1 par niveau du système complet : c’est l’étape d’évaluation annoncée en 4.3.3. Le script correspondant (ai/training/evaluate_v2.py) interroge le serveur d’inférence sur ces mêmes 100 cas et produit directement ces indicateurs ; il ne demande que l’accès au GPU sur lequel le modèle est servi. Au regard des 38 % de l’algorithme de repli, et de la précision au niveau du token déjà observée à l’entraînement, nous attendons du modèle un gain net, en particulier sur le taux de sous-triage, qui est la métrique de sécurité déterminante.

4.6 Validation fonctionnelle de bout en bout

Les mesures précédentes évaluent le modèle isolément. Restait à vérifier que la chaîne complète (interface, backend, file temps réel, modèle, validation médicale) fonctionne comme un tout, avec ses cinq rôles utilisateurs et ses échanges en temps réel. Nous l'avons fait en déroulant des scénarios complets sur le système déployé en production, dans sa configuration réelle : l'application hébergée sur le VPS interroge le serveur GPU du CRI à travers un réseau privé virtuel (VPN) sécurisé, exactement comme elle le ferait en service. Voici le déroulé de l'un de ces scénarios, tel que nous l'avons observé.

L'agent d'accueil se connecte avec son compte et enregistre un nouveau patient : identité, âge, motif de consultation tel que le patient l'exprime. Le système crée la visite et délivre un ticket. Le patient passe ensuite au poste de l'infirmier, qui saisit les constantes vitales (température, tension, fréquences cardiaque et respiratoire, saturation, niveau de douleur) et coche les symptômes observés, en complétant au besoin par du texte libre.

À la validation du formulaire, le backend assemble le dossier et interroge BetterEMS-Triage. Environ quatre secondes plus tard, la réponse s'affiche : niveau 2, ORANGE, avec le délai cible de dix minutes du protocole de Manchester, deux diagnostics probables accompagnés de leurs codes CIM-10, la liste des examens recommandés en premier recours, et une explication rédigée qui résume le raisonnement : l'infirmier sait non seulement quoi, mais pourquoi. Le patient est inséré dans la file d'attente à la position que son niveau lui confère, devant les cas JAUNE et VERT arrivés avant lui.

La suite se joue sur les écrans. La file du médecin se met à jour en temps réel, le cas ORANGE en évidence. Lorsque le médecin appelle le patient, l'écran TV de la salle d'attente affiche le numéro de ticket et le box d'affectation, accompagnés d'un signal sonore. En consultation, le médecin dispose du dossier complet et du triage proposé ; il le confirme d'un clic, ou le corrige, auquel cas son retour est enregistré et alimentera le ré-entraînement périodique du modèle. Dans notre scénario, le médecin a confirmé le niveau ORANGE. Chaque action du parcours (enregistrement, saisie des constantes, triage, appel, confirmation) est par ailleurs tracée avec son auteur et son rôle dans le journal d'audit du système, exigence élémentaire pour un outil qui touche à des décisions de soin.

Les captures d'écran de chaque étape de ce scénario figurent en annexe. Ce qu'elles attestent tient en une phrase : le système fonctionne en conditions de production, de la porte d'entrée du service jusqu'à la décision médicale, sans intervention technique à aucune étape.

4.7 Bilan et perspectives

Au terme de cette campagne expérimentale, le bilan se mesure à des résultats concrets. Notre modèle BetterEMS-Triage, adapté sur notre corpus propriétaire de 12 400 cas cliniques, atteint ≈ 96 % de précision en validation et rend ses décisions en 2 à 4 secondes par patient, une latence

pleinement compatible avec le rythme d'un service d'urgences. Sa sortie JSON structurée est consommée telle quelle par le backend : niveau, délai, diagnostics codés en CIM-10, examens et explication s'intègrent dans la file de priorité sans aucune couche d'interprétation intermédiaire. Et l'ensemble tourne sur un seul GPU, le modèle quantifié en Q4_K_M ne mobilisant qu'une fraction de la mémoire disponible en service.

Ce dernier point porte une conséquence qui dépasse la performance : la souveraineté des données. Tout le pipeline (saisie, triage, file, validation) s'exécute sur une infrastructure maîtrisée par l'établissement ; aucune donnée de santé ne transite par un service d'IA externe ni ne quitte le périmètre de l'hôpital. Pour des données aussi sensibles que des dossiers d'urgences, ce déploiement local n'est pas un détail d'architecture mais une condition d'acceptabilité, que les solutions de triage adossées à des API cloud étrangères ne peuvent pas offrir.

La campagne a aussi produit un enseignement méthodologique : la comparaison contrôlée des deux stratégies de fine-tuning a établi que, sur un corpus spécialisé de cette taille, la quantification 4 bits de QLoRA introduit une instabilité numérique rédhibitoire là où le LoRA fp16 converge sans incident. Un résultat reproductible, documenté sur trois taux d'apprentissage, et utile à quiconque adapte un modèle médical dans des conditions matérielles comparables.

Enfin, le positionnement par rapport à l'état de l'art du chapitre 2 est favorable. Les travaux de référence sur le triage par apprentissage automatique [32] [34] s'arrêtent à la classification hors ligne sur données rétrospectives ; les approches par traitement du langage [35] traitent le texte infirmier mais ne produisent pas de décision structurée intégrée à un flux de soins. Parmi les travaux passés en revue, BetterEMS est le seul système de bout en bout effectivement déployé : du formulaire d'admission à l'écran d'appel, en passant par un modèle de langage spécialisé servi en production.

Ces résultats ouvrent une feuille de route déjà engagée. L'évaluation détaillée par niveau (matrice de confusion 5×5 et F1 par classe sur les 100 cas de test) est en cours, avec un protocole défini. Un pilote clinique est planifié avec un service d'urgences partenaire de la région de Saïda ; il fournira la validation en conditions réelles que les cas de test ne peuvent pas remplacer. La boucle de feedback médecin, déjà opérationnelle dans le système, alimentera l'enrichissement progressif du corpus par des cas réels confirmés ou corrigés, en complément de nos cas synthétiques. L'extension au texte libre en arabe dialectal, fréquent dans l'expression des motifs de consultation en Algérie, est inscrite au plan de travail du prochain cycle d'entraînement.

4.8 Conclusion

Ce chapitre a rendu compte de la campagne expérimentale qui a transformé l'architecture du chapitre 3 en un système mesuré et déployé. La démarche comparative entre QLoRA et LoRA fp16 a identifié la configuration d'entraînement adaptée à notre corpus et à notre matériel ; l'entraînement de BetterEMS-Triage a convergé vers $\approx 96\%$ de précision en validation, avec une sélection de checkpoint qui a neutralisé le surapprentissage exactement comme le protocole

le prévoyait ; l'inférence quantifiée tient le temps réel des urgences sur un GPU unique; et la validation de bout en bout a montré le système au travail, de l'admission d'un patient à la confirmation de son triage par le médecin.

Un système qui fonctionne en production, un modèle propriétaire, un brevet en cours de dépôt : ces résultats ne sont pas seulement académiques. Ils constituent le socle d'un projet de startup, labellisé par l'incubateur de l'Université de Saïda. Le chapitre suivant change donc de registre et examine BetterEMS sous l'angle entrepreneurial : le problème de marché qu'il adresse, sa proposition de valeur, son modèle économique et sa feuille de route vers les premiers déploiements hospitaliers.

Étude entrepreneuriale : de la recherche à la startup

5.1 Introduction

Ce mémoire a un double statut. Il constitue d'abord un travail académique de Master, dont les quatre premiers chapitres ont posé les fondements théoriques, conçu le système et présenté les résultats expérimentaux. Il porte aussi un projet de startup : BetterEMS a été labellisé par l'incubateur de l'Université de Saïda dans le cadre du décret ministériel 1275, dit « un diplôme, une startup », qui permet à un étudiant de soutenir son projet de fin d'études sous la forme d'un projet d'entreprise innovante. Un brevet d'invention est par ailleurs en cours de dépôt pour protéger le procédé développé.

Ce double statut impose un chapitre d'un genre différent. Les chapitres précédents jugeaient le système à l'aune de la rigueur scientifique : qualité du dataset, stabilité de l'entraînement, performances d'inférence. Celui-ci le considère comme un produit destiné à de futurs clients, en l'occurrence des établissements de santé. Le passage de l'un à l'autre n'est pas qu'un changement de vocabulaire : il oblige à répondre à des questions que la démarche académique ne pose pas. Qui paierait pour cette solution ? Contre qui ou contre quoi se positionne-t-elle ? Comment survivrait-elle au-delà de la soutenance ?

Pour y répondre, nous suivons le canevas officiel du mémoire de projet de startup () : résumé du projet et présentation de l'équipe, problématique ciblée, solution et prototype, valeur ajoutée, analyse du marché, analyse concurrentielle, modèle économique, feuille de route et Lean Canvas. Nous nous sommes imposé une règle tout au long du chapitre : conserver la même exigence de rigueur que dans les chapitres expérimentaux. Aucun chiffre de marché n'est avancé sans source, chaque valeur défendue est déjà démontrée par le prototype, et ce qui relève des étapes suivantes, le pilote clinique en premier lieu, figure à sa place dans la feuille de route.

5.2 Résumé du projet et équipe

BetterEMS est un système d'aide au triage des urgences hospitalières destiné aux établissements de santé algériens. Le problème adressé est connu de tout service d'urgences : une file d'attente gérée par ordre d'arrivée ne distingue pas un cas bénin d'un cas vital, et le triage manuel, quand il existe, varie d'un soignant à l'autre. Notre solution est une application web couvrant tout le parcours du patient (enregistrement, prise des constantes, évaluation par notre modèle médical spécialisé BetterEMS-Triage, file d'attente priorisée, validation par le médecin), avec

un affichage de salle d'attente qui appelle les patients automatiquement. Le classement suit les cinq niveaux du protocole de Manchester [6] et chaque proposition de l'intelligence artificielle reste soumise à la décision du médecin. L'impact attendu est une prise en charge plus rapide des cas graves, un triage plus homogène et des indicateurs de pilotage dont les services manquent aujourd'hui. L'ensemble fonctionne sur l'infrastructure de l'établissement, sans transfert de données vers un service externe.

Le projet est porté par une équipe resserrée et complémentaire, dont les rôles se sont définis naturellement au fil du travail. Boutaleb Imane, étudiante en Master 2 Intelligence Artificielle et Sciences des Données, a conçu et développé le système : constitution du dataset de triage, fine-tuning du modèle de langage, développement du backend et du frontend. Merzoug Rayane, partenaire externe du projet, a pris en charge l'infrastructure : hébergement, mise en production et exploitation de la plateforme de démonstration. Rahmani Mohamed Elhadi, enseignant-chercheur et encadreur du mémoire, a assuré le suivi scientifique et méthodologique, du choix du modèle de base à la lecture critique des résultats. Le projet a en outre bénéficié de l'appui technique du Centre de Recherche en Informatique (CRI) de l'Université de Saïda, sous la responsabilité de M. Baki Abdtouab, qui a mis à disposition les moyens de calcul GPU sans lesquels le fine-tuning n'aurait pas été possible.

Tab. 5.1 : Équipe du projet BetterEMS et répartition des rôles.

Membre	Qualité	Rôle dans le projet
Boutaleb Imane	Étudiante M2, IA et Sciences des Données	Conception et développement du système et du modèle (dataset, fine-tuning, application)
Merzoug Rayane	Partenaire externe	Infrastructure, hébergement et mise en production
Rahmani Mohamed Elhadi	Enseignant-chercheur, encadreur	Suivi scientifique et méthodologique
M. Baki Abdtouab (CRI)	Appui technique institutionnel	Moyens de calcul GPU du CRI de l'Université de Saïda

Cette configuration (une étudiante au développement, un partenaire à l'exploitation, un encadreur à la méthode et un centre de recherche en soutien) correspond à l'esprit du décret 1275, qui encourage la transformation d'un travail universitaire en projet d'entreprise sans attendre la constitution d'une équipe complète.

5.3 Problématique ciblée

Le chapitre 1 a décrit le problème du triage sous l'angle théorique, par la théorie des files d'attente. Ici, le même problème est regardé du point de vue du client : un hôpital public, et plus précisément la direction d'un établissement et le chef de son service d'urgences. Pour eux, le triage défaillant n'est pas une question de discipline de file ; c'est un risque opérationnel, juridique et humain.

Quatre problèmes concrets ressortent de nos échanges préparatoires et de la littérature sur l'engorgement des urgences [10]. Le premier est l'attente excessive des cas urgents lorsque la file est gérée par ordre d'arrivée : un patient grave peut patienter derrière des cas bénins, avec un risque vital direct. Le deuxième est la variabilité du triage manuel : deux soignants peuvent classer différemment le même patient, et la fiabilité inter-évaluateurs des échelles de triage, même éprouvées, dépend fortement de la formation et de l'expérience [7]. Le troisième est la surcharge cognitive du personnel en période d'affluence : trier vite, sous pression, en gérant simultanément l'accueil et les soins, épuise les équipes et dégrade la qualité des décisions. Le quatrième, moins visible mais réel, est l'absence d'indicateurs : la plupart des services ne disposent d'aucune mesure fiable des temps d'attente, des volumes par niveau de gravité ou des heures de pointe, ce qui rend tout pilotage impossible.

Tab. 5.2 : Problèmes identifiés chez le client cible, avec leur impact et leur fréquence.

Problème identifié	Impact (Faible / Fort)	Fréquence (Rare / Fréquent)
Attente excessive des cas urgents en file FIFO	Fort	Fréquent
Variabilité du triage manuel entre soignants	Fort	Fréquent
Surcharge cognitive du personnel en période d'affluence	Fort	Fréquent
Absence d'indicateurs pour piloter le service	Moyen	Fréquent

Les points de douleur qui en découlent dépassent le cadre médical. Un cas grave non détecté à temps expose l'établissement à un risque médico-légal. L'épuisement du personnel alimente l'absentéisme et les départs. Les scènes de tension en salle d'attente, largement relayées, abîment l'image du service public de santé. Les conséquences se mesurent en temps perdu, en stress pour les soignants comme pour les patients et, dans les pires cas, en pertes de chances vitales.

Reste la question que pose le canevas : ce problème est-il assez important pour qu'un client paie pour une solution ? Nous pensons que oui, pour une raison simple : les trois premiers

problèmes du tableau 5.2 sont à la fois forts et fréquents, et ils touchent précisément ce que la responsabilité d'un directeur d'établissement couvre, à savoir la sécurité des patients et le fonctionnement du service. La confirmation définitive ne pourra toutefois venir que du terrain, et c'est l'objet du pilote prévu dans notre feuille de route (section 5.9).

5.4 Solution proposée et prototype

À chacun des quatre problèmes identifiés, BetterEMS apporte une réponse précise. À la file FIFO, le système substitue une file à priorité fondée sur les cinq niveaux du Manchester Triage System, avec des délais cibles affichés et un reclassement automatique : un cas rouge passe devant la file quel que soit son ordre d'arrivée. À la variabilité du triage manuel, il oppose une évaluation standardisée : notre modèle BetterEMS-Triage, un modèle de 7,24 milliards de paramètres obtenu par adaptation (fine-tuning LoRA) de BioMistral-7B [23] sur notre corpus propriétaire de plus de 12 000 cas de triage en français, applique les mêmes critères à chaque patient, et le médecin garde le dernier mot. Il confirme ou corrige chaque proposition, et chaque correction est enregistrée pour améliorer le modèle. À la surcharge cognitive, le système répond par la rapidité de la pré-analyse, deux à quatre secondes par patient sur le serveur GPU, et par l'automatisation des tâches périphériques : l'écran de salle d'attente appelle lui-même les patients, avec numéro de ticket, box assigné et signal sonore. À l'absence d'indicateurs, enfin, répond un tableau de bord qui agrège en temps réel les volumes, les niveaux de gravité et les temps d'attente, assorti d'un journal d'audit qui trace chaque action et son auteur.

Le prototype n'est pas une maquette. Il s'agit d'une application web fonctionnelle, déployée en production sur un serveur applicatif relié au serveur GPU du CRI par un réseau privé virtuel sécurisé. Le flux complet y est opérationnel de bout en bout : un agent d'accueil enregistre le patient et imprime son ticket ; un infirmier saisit les constantes vitales et les symptômes ; le modèle produit sa proposition de triage ; le patient apparaît dans la file priorisée ; le médecin l'appelle depuis son poste, l'écran de salle d'attente l'affiche, et la consultation se conclut par la validation ou la correction du triage. Un rôle d'ambulancier permet en outre le pré-enregistrement d'un patient pendant son transport. Cinq rôles d'utilisateurs, neuf entités métier, une file mise à jour en temps réel : le périmètre couvert est celui d'un produit, pas d'une démonstration partielle. Une instance de démonstration est accessible en ligne : [\[lien de démonstration\]](#). Les captures d'écran des principales interfaces sont regroupées en annexe.

Tout ce que nous décrivons ici a été exécuté réellement sur le prototype : le flux de prise en charge, le triage par le modèle, la file temps réel, l'écran d'appel, le mécanisme de repli par règles lorsque le modèle est indisponible. L'évaluation de la performance clinique en conditions réelles constitue, elle, la première étape de la feuille de route, avec un pilote planifié dans un service d'urgences partenaire (section 5.9) : un produit de santé ne se vend pas sur des intentions.

5.5 Valeur ajoutée

Notre proposition de valeur tient en une phrase : un triage standardisé, assisté par IA et validé par le médecin, en moins de quatre secondes, déployable entièrement sur l'infrastructure de l'hôpital. Le canevas officiel conseille de ne défendre en détail que les valeurs réellement validées par le prototype, quitte à n'en retenir qu'une. Nous en argumentons deux, parce que le prototype les démontre toutes deux.

La première est la réduction du risque. Le danger principal d'un service d'urgences saturé est le cas grave qui attend. BetterEMS réduit ce risque par trois mécanismes complémentaires, tous opérationnels dans le prototype. La priorisation est systématique : aucun patient n'entre dans la file sans niveau de gravité, et l'ordre de passage en découle mécaniquement. L'évaluation est standardisée : les mêmes critères s'appliquent au même tableau clinique, ce qui atténue la variabilité entre évaluateurs constatée dans le triage manuel [7]. Et la supervision humaine est structurelle, non décorative : le médecin valide ou corrige chaque triage, et un algorithme de repli à base de règles prend le relais si le modèle est indisponible ou si sa confiance est jugée insuffisante. Lors de nos essais sur des cas de test construits manuellement, le système s'est montré particulièrement sûr sur la détection des cas critiques (niveaux rouge et orange), ce qui est exactement la propriété recherchée; le pilote clinique de la feuille de route permettra de la quantifier sur des patients réels.

La seconde est la souveraineté des données. Les données de santé comptent parmi les plus sensibles qui soient, et leur gouvernance est devenue une exigence internationale autant que nationale [30]. La plupart des solutions d'IA médicale reposent sur des interfaces de programmation hébergées à l'étranger : chaque patient trié signifie alors des données cliniques transmises hors du pays, vers un prestataire soumis à un droit étranger. Notre architecture élimine ce problème par construction. Le modèle, quantifié pour le déploiement, s'exécute sur un unique serveur GPU et sert ses réponses en deux à quatre secondes ; l'ensemble du système (base de données, application, modèle) peut être installé dans l'enceinte de l'établissement. Aucune donnée patient ne quitte l'infrastructure de l'hôpital. Cette propriété n'est pas un argument commercial plaqué après coup : elle a orienté nos choix techniques dès le départ, du modèle ouvert au format d'inférence léger.

Le gain de temps (une évaluation en quelques secondes là où un triage manuel complet en demande plusieurs minutes) figure dans notre proposition de valeur, mais nous le considérons comme une conséquence des deux axes précédents plutôt que comme un argument autonome : c'est parce que l'évaluation est rapide que la standardisation et la priorisation deviennent praticables sans ralentir l'accueil.

5.6 Analyse du marché cible

Notre client principal n'est pas un consommateur mais une institution. Le marché visé est d'abord de type B2G (*Business to Government*) : les hôpitaux publics algériens, centres hospitalo-universitaires (CHU) et établissements publics hospitaliers (EPH), qui concentrent l'essentiel de l'activité d'urgence du pays. C'est là que les problèmes du tableau 5.2 sont les plus aigus, et c'est là que la décision d'achat relève de la commande publique. Cette nature publique du client impose une contrainte que notre modèle économique devra intégrer : le délai de paiement d'un marché public se compte en mois, de l'ordre de 120 jours entre la livraison et l'encaissement, ce qui pèse directement sur la trésorerie d'une jeune structure.

La structure du système hospitalier public algérien dessine trois segments, que nous classons par taille décroissante et par ordre de priorité commerciale. Le pays compte d'abord une quinzaine de CHU, grands établissements universitaires, peu nombreux mais prescripteurs : équiper un CHU sert de référence de prestige et facilite l'accès aux autres établissements ; nous visons environ 20 % de nos ventes sur ce segment. Viennent ensuite près de 200 hôpitaux de wilaya (les EPH), qui forment le cœur de notre cible et sur lesquels nous projetons la moitié de nos ventes. Le troisième segment réunit les petits établissements et les polycliniques dotés d'un accueil d'urgences, vers lesquels nous orientons les 30 % restants. Cette répartition, 20 / 50 / 30, structure à la fois notre effort commercial et la grille tarifaire par palier présentée à la section 5.8.

La segmentation pertinente est simple, car le produit s'adresse à un type de service bien identifié. Trois critères suffisent : la taille et le statut de l'établissement (CHU, hôpital de wilaya ou petit établissement), qui déterminent à la fois le circuit d'achat et le palier tarifaire ; l'existence d'un service d'urgences structuré, qui conditionne l'utilité du produit ; et le niveau d'équipement informatique, qui conditionne la facilité du déploiement. Un CHU disposant déjà d'un réseau interne et d'un personnel informatique constitue le terrain le plus favorable ; un petit établissement faiblement équipé demandera une prestation d'installation plus lourde.

Nos *early adopters* sont identifiés : les services d'urgences des établissements universitaires de la région de Saïda, avec lesquels l'université et le CRI entretiennent des relations institutionnelles directes. Ce choix n'est pas par défaut. Un service hospitalo-universitaire est habitué à accueillir des projets de recherche, dispose de personnel médical susceptible de s'impliquer dans une évaluation clinique, et la proximité géographique permet à notre équipe d'assurer un accompagnement quotidien pendant la phase pilote. C'est aussi le cadre naturel pour l'évaluation formelle dont le projet a besoin avant toute diffusion plus large.

5.7 Analyse concurrentielle

La concurrence de BetterEMS ne ressemble pas à celle d'une application grand public : notre principal concurrent n'est pas une entreprise, c'est l'habitude. Le tableau 5.3 recense les trois

alternatives auxquelles un établissement peut recourir aujourd'hui.

Tab. 5.3 : Concurrents directs et indirects de BetterEMS.

Concurrent	Type	Limites de la solution
Processus manuel (registre papier, ordre d'arrivée)	Indirect	Gratuit mais non standardisé ; aucune traçabilité ; pas d'indicateurs ; dépend entièrement de la disponibilité et de l'expérience du personnel
Systèmes d'information hospitaliers et dossiers patients informatisés génériques	Indirect	Couvrent la gestion administrative (admissions, facturation, dossiers) mais n'offrent pas de triage intelligent ni de file priorisée temps réel
Solutions internationales de triage assisté par IA	Direct	Non localisées (langue, pratiques cliniques, contexte algérien) ; coût de licence élevé ; souvent hébergées sur un cloud étranger, ce qui pose le problème de la souveraineté des données de santé

Le processus manuel reste l'alternative dominante : il ne coûte rien à court terme, ce qui en fait un concurrent redoutable malgré toutes ses limites. Les systèmes d'information hospitaliers présents en Algérie traitent l'administratif, pas la décision clinique d'orientation ; ils sont pour nous des compléments potentiels plus que des rivaux. Les solutions internationales de triage assisté, dont la littérature scientifique atteste la maturité technique [32, 35], constituent la concurrence directe ; mais aucune, à notre connaissance, n'est localisée pour le contexte algérien, et leur dépendance fréquente à un hébergement étranger se heurte frontalement à la sensibilité des données de santé.

L'analyse SWOT qui suit nomme les faiblesses du projet, comme le canevas officiel le demande ; chacune est datée et assortie de la réponse prévue dans la feuille de route.

Tab. 5.4 : Analyse SWOT du projet BetterEMS.

Élément	Analyse pour BetterEMS
Forces (S)	Prototypage fonctionnel de bout en bout, déployé et démontrable ; modèle spécialisé exécutable sur un seul GPU ; localisation en français et pour le contexte algérien ; validation médicale intégrée au flux ; appui institutionnel de l'université et du CRI
Faiblesses (W)	Validation clinique en milieu hospitalier : pilote planifié en première phase de la feuille de route ; équipe resserrée de trois personnes, aux profils complémentaires ; communication commerciale non encore lancée à ce stade ; dataset d'entraînement synthétique, que les retours médicaux du pilote viendront enrichir
Opportunités (O)	Numérisation engagée du secteur de la santé ; dispositif « un diplôme, une startup » et écosystème d'incubation ; absence de solution localisée équivalente sur le marché national
Menaces (T)	Lenteur des procédures de marchés publics ; exigences réglementaires propres aux outils logiciels en santé ; arrivée possible d'acteurs étrangers disposant de moyens supérieurs

Notre différenciation découle directement de ce tableau. Face au processus manuel, nous apportons la standardisation et la traçabilité qui lui manquent, sans retirer au médecin sa décision. Face aux solutions internationales, nous jouons sur trois terrains où elles sont structurellement désavantagées. L'ancrage local, d'abord, avec une équipe sur place, une interface en français et un dataset construit autour des pathologies et des pratiques du contexte algérien. La souveraineté, ensuite, puisque aucune donnée patient ne sort de l'établissement, là où une solution cloud étrangère ne peut tout simplement pas offrir cette garantie. Le coût d'infrastructure, enfin, car un unique GPU suffit à servir un service d'urgences, là où les solutions concurrentes supposent des abonnements récurrents à des services d'inférence externes. Cette différenciation définit le terrain sur lequel une équipe resserrée peut légitimement se battre.

5.8 Modèle économique

Le modèle de revenus retenu est hybride, choisi pour sa compatibilité avec la commande publique : il associe une licence initiale payée une seule fois, un abonnement annuel récurrent, et des modules additionnels facturés à la carte. La licence initiale couvre le déploiement, l'intégration au système d'information hospitalier (SIH) et la première année de service. L'abonnement annuel finance ensuite le support, les mises à jour continues du modèle d'intelligence artificielle et une hotline ; fixé à environ 40 % de la licence, il transforme un premier contrat en revenu récurrent prévisible. Ce couple licence plus abonnement répond au fonctionnement du marché visé : un établissement public ne peut généralement pas acheter un service hébergé à l'étranger pour ses données de patients, mais il peut inscrire à son budget une licence logicielle et un contrat de maintenance annuel, objets familiers de la commande publique. Le modèle économique et l'argument de souveraineté de la section 5.5 sont donc les deux faces d'une même pièce.

Les tarifs sont déclinés en trois paliers, alignés sur les segments de la section 5.6 (CHU, hôpital de wilaya, petit établissement), afin que le prix suive la taille et les moyens du client :

- CHU : licence initiale de 1 500 000 DA, abonnement annuel de 600 000 DA.
- Hôpital de wilaya (EPH) : licence de 1 200 000 DA, abonnement de 480 000 DA par an.
- Petit établissement ou polyclinique : licence de 900 000 DA, abonnement de 360 000 DA par an.

Compte tenu de la répartition des ventes (20 / 50 / 30), la licence moyenne pondérée ressort à 1 170 000 DA et l'abonnement moyen pondéré à 468 000 DA par an. À ces deux revenus de base s'ajoutent quatre sources complémentaires. La formation d'onboarding est facturée 250 000 DA par hôpital, l'adhésion du personnel (agents d'accueil, infirmiers, médecins) conditionnant tout le projet. La personnalisation, demandée par environ 30 % des établissements, est facturée de l'ordre de 300 000 DA. Un module premium, vendu 150 000 DA par an et adopté par près de 40 % des sites, illustre une stratégie de type *land and expand* : nous entrons par l'offre de base, puis nous élargissons le périmètre au fil de l'usage. Enfin, nous proposons la revente du serveur GPU nécessaire à l'inférence à 520 000 DA, pour un coût d'achat de 400 000 DA, soit une marge de 120 000 DA ; environ 80 % des hôpitaux choisissent de l'acquérir auprès de nous plutôt que de s'approvisionner eux-mêmes. La fidélité du modèle récurrent est élevée : nous retenons un taux de renouvellement de l'abonnement de 95 % par an, soit un taux d'attrition (*churn*) de seulement 5 %.

Un point mérite d'être explicité : nous excluons toute monétisation des données. Les données traitées par BetterEMS (motifs de consultation, constantes vitales, hypothèses diagnostiques) sont des données de santé, c'est-à-dire des données sensibles au sens de la réglementation sur la protection des données personnelles, et leur exploitation commerciale serait à la fois illégale, contraire à l'éthique médicale [31] et destructrice de la confiance sur laquelle repose notre

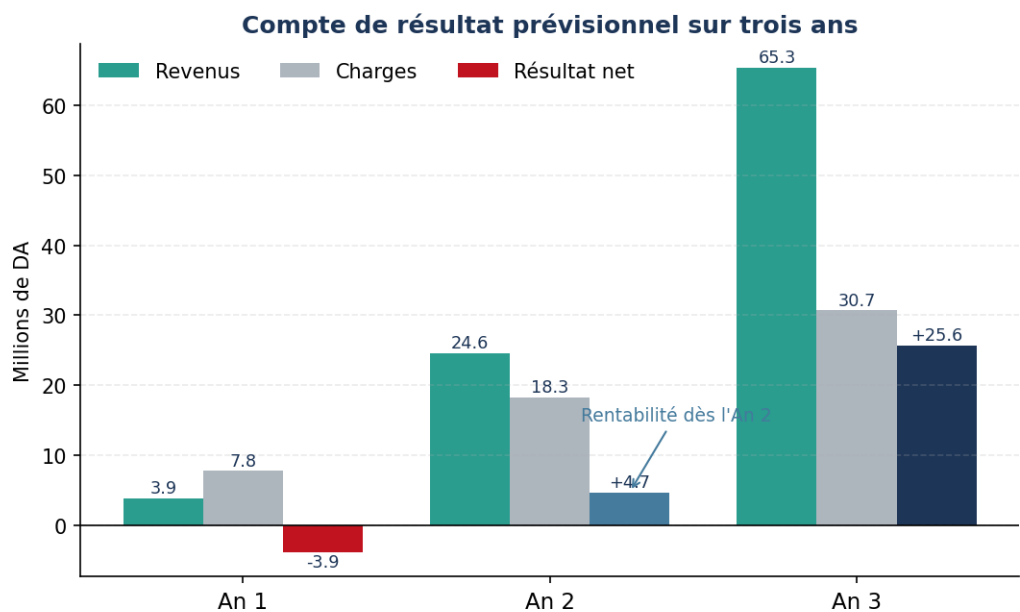


Fig. 5.1 : Compte de résultat prévisionnel sur trois ans (modèle B2G récurrent).

proposition de valeur. Les retours des médecins servent exclusivement à améliorer le modèle pour l’établissement, dans le cadre de la boucle d’apprentissage décrite au chapitre 3.

Côté coûts, la structure reste légère par conception. Chaque déploiement entraîne un coût d’installation de 150 000 DA par hôpital. Le poste principal est la masse salariale chargée (incluant 35 % de cotisations CNAS), qui croît avec l’équipe : environ 5,02 M DA la première année (les deux cofondateurs), 11,34 M DA la deuxième (quatre personnes) et 15,23 M DA la troisième (cinq personnes), avec une inflation des charges de 5 % par an. Le serveur GPU revendu au client, déjà évoqué, constitue un coût variable refacturé avec marge plutôt qu’une charge propre.

La projection financière sur trois ans repose sur un rythme d’acquisition réaliste : 3 hôpitaux la première année (dont un pilote gratuit, le CHU de Saïda), 12 la deuxième et 30 la troisième. La figure 5.1 résume le compte de résultat prévisionnel qui en découle, détaillé au tableau 5.5.

Tab. 5.5 : Compte de résultat prévisionnel sur trois ans (DA).

Indicateur	An 1	An 2	An 3
Hôpitaux acquis dans l’année	3	12	30
Total des revenus (DA)	3 852 000	24 625 800	65 349 000
Total des charges (DA)	7 752 000	18 282 000	30 695 100
Résultat net (DA)	−3 900 000	+4 694 412	+25 643 886
Marge nette	n.s. (perte)	≈ 19 %	≈ 39 %

La lecture du tableau confirme la trajectoire attendue d’une startup à revenu récurrent. La première année est déficitaire, à environ −3,9 M DA, le temps d’amorcer la base installée. Dès

la deuxième année, l'effet d'accumulation des abonnements rend le projet bénéficiaire, avec un résultat net de l'ordre de 4,7 M DA et une marge nette proche de 19 %. La troisième année, le cumul des licences nouvelles et des abonnements renouvelés porte le résultat net à environ 25,6 M DA, pour une marge nette voisine de 39 %. Ces résultats s'entendent avant impôt sur les bénéfices des sociétés (IBS), au taux de 26 %.

Plusieurs indicateurs de viabilité confirment la solidité du modèle. Le seuil de rentabilité se situe à 9 hôpitaux vendus par an : il est franchi dès la deuxième année, qui en compte 12. La marge brute dégagée par hôpital atteint 1 456 000 DA (un revenu de première année de 1 926 000 DA, diminué d'un coût variable de 470 000 DA). Surtout, le rapport entre la valeur vie client et le coût d'acquisition (ratio LTV/CAC) s'établit à 22,8, très au-dessus des seuils usuels (un ratio est jugé bon au-delà de 3 et excellent au-delà de 5) : l'effet conjugué d'un abonnement reconduit à 95 % et d'un coût d'acquisition contenu explique cette valeur, caractéristique d'un modèle récurrent sain.

Reste la question du financement de la première année, structurellement déficitaire, aggravée par le délai de paiement public de 120 jours qui gonfle le besoin en fonds de roulement. Nous prévoyons une levée d'amorçage (*seed*) de 12 M DA, dimensionnée pour absorber la perte de l'An 1 et ce décalage de trésorerie le temps que les abonnements prennent le relais. Les sources visées relèvent toutes de l'écosystème national de soutien à l'innovation : l'incubateur de l'Université de Saïda, l'Agence nationale d'appui et de développement de l'entrepreneuriat (ANADE), Algeria Venture et l'Algeria Startup Challenge.

5.9 Feuille de route

La feuille de route traduit une conviction déjà exprimée : rien ne sert de vendre avant d'avoir validé cliniquement. Elle s'étale sur trois ans, fait de l'évaluation formelle le verrou de la première année, puis cale son rythme commercial sur la trajectoire financière de la section 5.8 : 3 hôpitaux la première année, 12 la deuxième et 30 la troisième.

Tab. 5.6 : Feuille de route du projet sur trois ans.

Phase	Activités	Résultats attendus
An 1 : Pilote et amorçage	Déploiement et évaluation clinique formelle au CHU de Saïda en pilote gratuit (matrice de confusion, F1 par niveau, accord médecin/IA); dépôt du brevet ; structuration juridique ; levée d’amorçage de 12 M DA ; signature des 2 premières ventes	Pilote opérationnel et rapport d’évaluation clinique ; 3 hôpitaux équipés ; financement seed bouclé
An 2 : Rentabilité	Industrialisation du déploiement et de l’onboarding ; réponse aux consultations publiques ; renforcement de l’équipe à quatre personnes ; déploiement des modules premium	12 hôpitaux acquis dans l’année ; seuil de rentabilité (9 hôpitaux) franchi ; résultat net positif
An 3 : Expansion nationale	Extension à l’échelle nationale au-delà de la région de Saïda ; intégration avec le SAMU (pré-enregistrement pendant le transport, déjà prototypé) ; négociation d’un accord-cadre avec le ministère de la Santé	30 hôpitaux acquis dans l’année ; présence nationale ; idéalement un accord-cadre ministériel

Les indicateurs de suivi associés à ce plan sont volontairement peu nombreux. Pour la première année, nous en retenons trois sur le plan clinique : un site pilote effectivement opérationnel ; le taux d’accord entre le médecin et la proposition de l’IA, que le système mesure nativement puisque chaque validation ou correction est enregistrée ; et la satisfaction du personnel utilisateur, recueillie par questionnaire auprès des agents, infirmiers et médecins du service pilote. À partir de la deuxième année, le suivi devient aussi commercial et financier : nombre d’hôpitaux acquis, taux de renouvellement des abonnements et résultat net, qui sont les variables dont dépend l’atteinte du seuil de rentabilité. Le taux d’accord médecin/IA conserve son double rôle : indicateur de qualité pour le déploiement, et matière première du ré-entraînement périodique du modèle.

Cette feuille de route prolonge un projet déjà réalisé. Le développement de BetterEMS

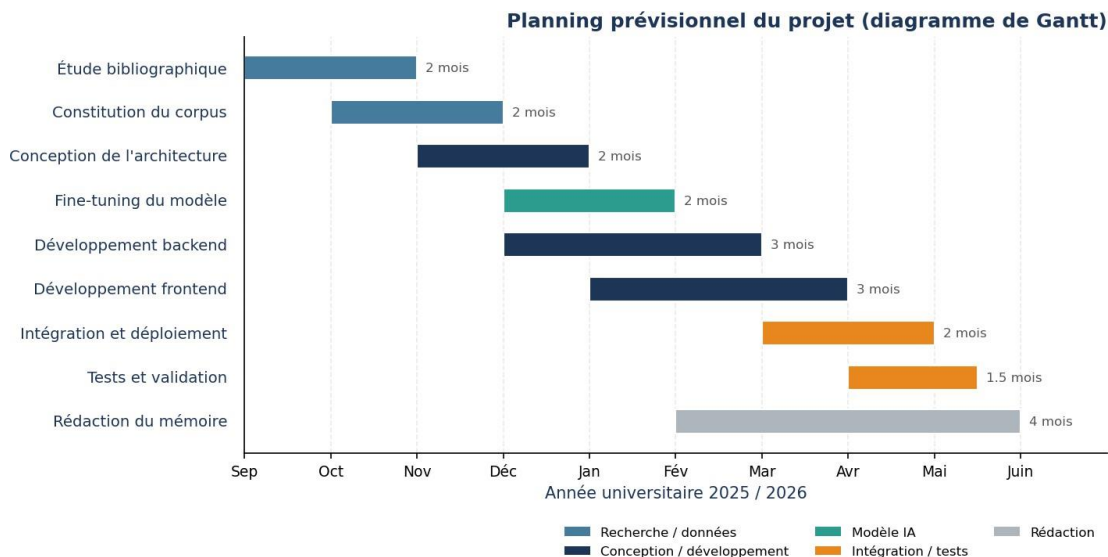


Fig. 5.2 : Diagramme de Gantt : planning de réalisation du projet.

Lean Canvas — BetterEMS

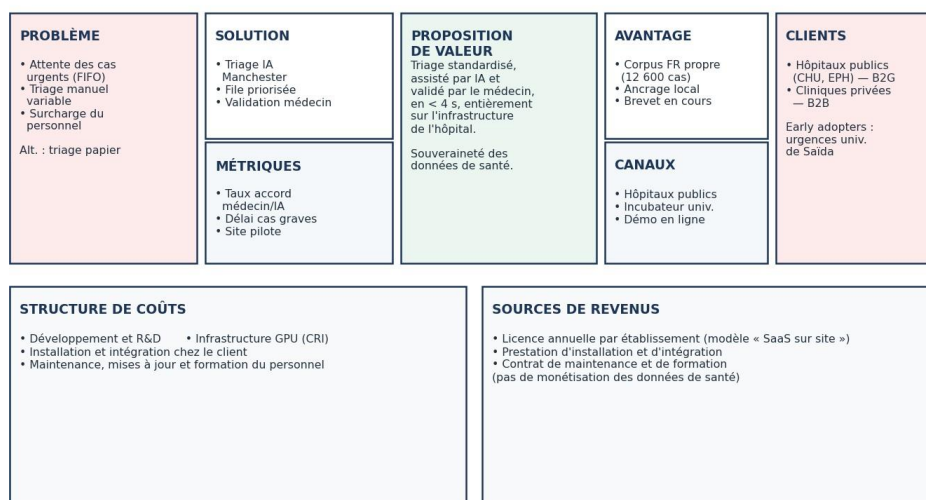


Fig. 5.3 : Lean Canvas du projet BetterEMS.

s'est lui-même déroulé sur l'année universitaire, de l'étude bibliographique à la rédaction du présent mémoire, en menant de front la constitution du corpus, l'adaptation du modèle et le développement du système. La figure 5.2 en donne le planning sous forme de diagramme de Gantt.

5.10 Lean Canvas

Le Lean Canvas de la figure 5.3 condense l'ensemble de l'étude sur une page, selon le format demandé par le canevas officiel.

Quelques commentaires suffisent, bloc par bloc, pour relier la figure aux sections précédentes. Le bloc *Problème* reprend les trois problèmes à fort impact du tableau 5.2 (attente des cas urgents, variabilité du triage, surcharge du personnel) et mentionne comme alternative existante le processus manuel, qui reste la pratique dominante. Le bloc *Segments de clients* désigne les hôpitaux publics (CHU, EPH) comme cœur de cible et les cliniques privées comme segment secondaire, avec pour *early adopters* les services d'urgences universitaires de la région de Saïda. La *Proposition de valeur unique* est celle de la section 5.5 : un triage standardisé, assisté par IA et validé par le médecin, en moins de quatre secondes, entièrement sur l'infrastructure de l'hôpital.

Le bloc *Solution* résume les trois fonctions essentielles : évaluation automatisée selon le protocole de Manchester, file d'attente priorisée en temps réel avec écran d'appel, et boucle de validation médicale qui alimente l'amélioration continue. Les *Métriques clés* sont celles de la feuille de route (taux d'accord médecin/IA, temps entre l'admission et le triage, satisfaction du personnel), toutes mesurables avec le prototype actuel. Les *Canaux* reflètent la réalité d'un marché institutionnel : l'incubateur et l'université comme premiers relais, le contact direct avec les directions d'établissement, et à terme les procédures de marchés publics.

L'*Avantage déloyal* (ce qui ne peut être ni copié ni acheté facilement) réside selon nous dans la combinaison du dataset de triage en français construit manuellement, de la boucle de retours médicaux qui l'enrichit à chaque déploiement, de l'ancrage institutionnel auprès du CRI et de l'université, et du brevet en cours de dépôt. La *Structure de coûts* identifie les postes principaux : masse salariale chargée (le poste dominant), coûts d'installation par hôpital, matériel GPU et besoin en fonds de roulement lié au délai de paiement public de 120 jours, couvert par une levée d'amorçage de 12 M DA. La *Structure de revenus*, enfin, reprend le modèle hybride de la section 5.8 : une licence initiale par établissement, un abonnement annuel récurrent (renouvelé à 95 %) qui constitue le revenu de fond, et des modules additionnels (formation, personnalisation, module premium, revente du serveur GPU) selon une logique de *land and expand*. C'est ce caractère récurrent qui porte le rapport LTV/CAC à 22,8 et fait passer le projet de la perte de l'An 1 à un résultat net d'environ 25,6 M DA en An 3.

5.11 Conclusion

L'exercice entrepreneurial aura forcé le projet académique à répondre à des questions qu'un mémoire classique laisse de côté : qui paie, contre quoi l'on se positionne, et ce qui sépare un prototype démontrable d'un produit déployable. En retour, la rigueur académique des chapitres précédents donne au discours de startup ce qui lui manque le plus souvent : des résultats mesurés, une évaluation clinique posée comme préalable et non comme formalité, et un refus constant des chiffres invérifiables. Les deux démarches, loin de se gêner, se sont renforcées : c'est précisément parce que le système existe et fonctionne que le projet d'entreprise peut être présenté sans promesses excessives.

Conclusion générale

Au départ de ce travail, il y avait un dysfonctionnement simple à énoncer et difficile à corriger : dans les services d'urgences algériens, l'ordre de passage des patients reflète davantage leur heure d'arrivée que la gravité de leur état. Le triage, lorsqu'il est pratiqué, dépend de l'expérience et de la disponibilité de l'infirmier d'accueil, et varie d'une équipe à l'autre, d'une heure de garde à l'autre. Nous avons fait le pari qu'un modèle de langage médical, correctement adapté au contexte local et strictement encadré par la validation du médecin, pouvait apporter à ce processus la standardisation qui lui manque sans rien retirer au jugement clinique. Ce mémoire a retracé la construction de cette réponse, du cadre théorique des files d'attente jusqu'au système déployé et à sa valorisation en startup.

La démarche a suivi un fil constant : partir du protocole de Manchester [6], le traduire en données, puis en modèle, puis en outil utilisable au guichet des urgences. Faute de corpus de triage en français adapté au terrain algérien, nous avons construit le nôtre. Douze mille six cents cas cliniques ont été rédigés à la main, couvrant la cardiologie, la traumatologie, la pédiatrie, les intoxications et les pathologies infectieuses courantes localement ; leur nettoyage a demandé 336 corrections, ce qui rappelle qu'un jeu de données ne vaut que par la rigueur de sa vérification. Le corpus a été réparti en 12 400 cas d'entraînement, 100 de validation et 100 de test, ces deux derniers ensembles étant tenus à l'écart de l'entraînement pour garantir une évaluation honnête.

L'adaptation du modèle a été l'étape la plus instructive du projet, parce qu'elle a exigé une véritable démarche expérimentale : observation, diagnostic, correction. Notre premier essai reposait sur QLoRA, séduisant sur le papier puisqu'il promettait de tenir l'entraînement dans une mémoire GPU réduite [25]. La perte a divergé dès les premiers pas, et le phénomène s'est reproduit à trois taux d'apprentissage différents : la cause n'était pas un réglage malheureux, mais l'instabilité numérique de la quantification sur 4 bits appliquée à un petit jeu de données spécialisé. Le passage à LoRA en précision fp16 [24], avec un rang réduit et un taux d'apprentissage prudent, a immédiatement stabilisé l'entraînement. Entraîné sur le serveur GPU du CRI de l'Université de Saïda, notre modèle BetterEMS-Triage, fort de 7,24 milliards de paramètres hérités de BioMistral-7B [23] dont environ 84 millions ajustés sur notre corpus, a atteint $\approx 96\%$ de précision (token accuracy) sur l'ensemble de validation, le meilleur point d'arrêt se situant dès la première époque; au-delà, le surapprentissage était manifeste, la perte d'entraînement continuant de chuter pendant que celle de validation remontait.

Un modèle isolé dans un environnement d'expérimentation n'aurait rien changé aux urgences. Nous l'avons donc fusionné, exporté, puis quantifié en Q4_K_M pour le déploiement, sans dégradation perceptible des réponses sur nos cas de test, et le serveur d'inférence rend sa proposition en deux à quatre secondes par patient. Autour de lui, BetterEMS couvre le parcours complet : enregistrement du patient, saisie des constantes et des symptômes par l'infirmier, proposition

de triage avec diagnostics probables et explication, file d'attente priorisée en temps réel, écran d'appel en salle d'attente ; cinq rôles d'utilisateurs et neuf entités métier structurent ce parcours. Chaque proposition passe entre les mains d'un médecin, qui la confirme ou la corrige ; ces retours sont conservés et alimenteront les ré-entraînements périodiques. Un algorithme de secours à base de règles prend le relais si le modèle est indisponible ou insuffisamment confiant. L'ensemble est déployé en production sur une infrastructure volontairement sobre, un serveur applicatif et le serveur GPU du CRI reliés par un réseau privé virtuel sécurisé, sans qu'aucune donnée patient ne sorte de ce périmètre.

De ce parcours, nous retenons quatre enseignements. Sur un petit jeu de données spécialisé, l'entraînement LoRA en fp16 s'est montré nettement plus stable que QLoRA en 4 bits : l'économie de mémoire promise par la quantification se paie en fragilité numérique, et mieux vaut un GPU correctement dimensionné qu'une compression agressive à l'entraînement. À l'inférence, en revanche, la quantification Q4_K_M s'est révélée un excellent compromis : elle réduit nettement la taille du modèle sans perte perceptible, ce qui met le déploiement à la portée d'un matériel raisonnable. Troisième leçon, un déploiement souverain et entièrement local d'un modèle de langage médical est réaliste avec les moyens d'une université algérienne ; l'objection de confidentialité opposée aux solutions hébergées à l'étranger trouve donc une réponse pratique, et non seulement de principe. Le dernier enseignement tient à la forme même du projet : mener de front le mémoire académique et la startup labellisée au titre du décret ministériel 1275, avec un brevet d'invention en cours de dépôt, a imposé une discipline de résultat dont les deux volets ont bénéficié, le prototype donnant de la crédibilité au dossier entrepreneurial et l'exigence académique garantissant que chaque résultat avancé est mesuré et vérifiable.

La feuille de route qui prolonge ce travail est déjà engagée. La prochaine étape planifiée est l'évaluation détaillée par niveau, matrice de confusion et scores F1 sur l'ensemble de test, avec une attention particulière aux sous-triages des cas graves : elle objectivera des observations qualitatives déjà nettes, excellente détection des cas rouges et oranges, quelques confusions aux frontières orange/jaune et vert/bleu. Viendra ensuite le pilote clinique dans un service d'urgences partenaire, qui mesurera le taux d'accord entre le modèle et les médecins sur des patients réels et alimentera la boucle d'apprentissage continu de corrections authentiques, enrichissant le corpus au-delà de sa base synthétique. À plus long terme, plusieurs extensions sont déjà esquissées :

- une application mobile de pré-enregistrement pour les ambulanciers, prolongée par une intégration avec la régulation du SAMU, afin que le triage commence avant l'arrivée à l'hôpital ;
- la prise en charge de l'arabe dialectal dans la saisie des motifs de consultation, au plus près de la langue réellement parlée par les patients ;
- des mécanismes d'explicabilité plus riches, pour que le médecin comprenne sur quels éléments le modèle fonde sa proposition ;
- l'interopérabilité avec les systèmes d'information hospitaliers existants, condition d'une adoption sans double saisie ;

- côté startup, le passage du prototype au premier site pilote, puis l'obtention des agréments et les premiers contrats avec des établissements publics.

La crise des urgences algériennes relève d'abord de moyens humains et d'organisation, et aucun logiciel n'y suffira seul. Ce mémoire démontre en revanche qu'un outil de triage porté par un modèle de langage de plusieurs milliards de paramètres, rédigé en français, entraîné et hébergé localement, est à la portée d'une équipe universitaire algérienne, et qu'il peut être conçu d'emblée pour laisser la décision finale au médecin, conformément aux recommandations internationales sur l'intelligence artificielle en santé [30]. Si le pilote clinique confirme ce que les premiers essais laissent entrevoir, quelques minutes gagnées sur la prise en charge de chaque cas grave justifieraient à elles seules l'effort consenti. C'est à cette aune que ce travail demande à être jugé.

Bibliographie

- [1] L. Kleinrock, *Queueing Systems, Volume 1 : Theory*. New York : Wiley-Interscience, 1975.
- [2] D. Gross, J. F. Shortle, J. M. Thompson et C. M. Harris, *Fundamentals of Queueing Theory*, 4^e éd. Hoboken : Wiley, 2008.
- [3] J. Medhi, *Stochastic Models in Queueing Theory*, 2^e éd. San Diego : Academic Press, 2002.
- [4] D. G. Kendall, « Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain », *Annals of Mathematical Statistics*, vol. 24, n^o 3, p. 338–354, 1953.
- [5] J. D. C. Little, « A proof for the queueing formula : $L = \lambda W$ », *Operations Research*, vol. 9, n^o 3, p. 383–387, 1961.
- [6] K. Mackway-Jones, J. Marsden et J. Windle, *Emergency Triage : Manchester Triage Group*, 3^e éd. Oxford : Wiley-Blackwell, 2014.
- [7] R. C. Wuerz, L. W. Milne, D. R. Eitel, D. Travers et N. Gilboy, « Reliability and validity of a new five-level triage instrument », *Academic Emergency Medicine*, vol. 7, n^o 3, p. 236–242, 2000.
- [8] R. Beveridge et al., « The Canadian Triage and Acuity Scale : a new and critical element in health care reform », *Journal of Emergency Medicine*, vol. 16, n^o 3, p. 507–511, 1998.
- [9] N. Gilboy, P. Tanabe, D. Travers et A. M. Rosenau, *Emergency Severity Index (ESI) : A Triage Tool for Emergency Department Care*, version 4. Rockville : Agency for Healthcare Research and Quality, 2020.
- [10] M. Christ, F. Grossmann, D. Winter, R. Bingisser et E. Platz, « Modern triage in the emergency department », *Deutsches Ärzteblatt International*, vol. 107, n^o 50, p. 892–898, 2010.
- [11] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York : Springer, 2006.
- [12] I. Goodfellow, Y. Bengio et A. Courville, *Deep Learning*. Cambridge : MIT Press, 2016.
- [13] T. M. Mitchell, *Machine Learning*. New York : McGraw-Hill, 1997.
- [14] S. Russell et P. Norvig, *Artificial Intelligence : A Modern Approach*, 4^e éd. Hoboken : Pearson, 2021.

- [15] Y. Goldberg, *Neural Network Methods for Natural Language Processing*. San Rafael : Morgan & Claypool, 2017.
- [16] D. Jurafsky et J. H. Martin, *Speech and Language Processing*, 3^e éd. (version préliminaire). Stanford University, 2024.
- [17] T. Mikolov, K. Chen, G. Corrado et J. Dean, « Efficient estimation of word representations in vector space », arXiv :1301.3781, 2013.
- [18] A. Vaswani et al., « Attention is all you need », in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [19] J. Devlin, M.-W. Chang, K. Lee et K. Toutanova, « BERT : Pre-training of deep bidirectional transformers for language understanding », in *NAACL-HLT*, 2019.
- [20] T. Brown et al., « Language models are few-shot learners », in *NeurIPS*, 2020.
- [21] H. Touvron et al., « LLaMA : Open and efficient foundation language models », arXiv :2302.13971, 2023.
- [22] A. Q. Jiang et al., « Mistral 7B », arXiv :2310.06825, 2023.
- [23] Y. Labrak, A. Bazoge, E. Morin, P.-A. Gourraud, M. Rouvier et R. Dufour, « BioMistral : A collection of open-source pretrained large language models for medical domains », in *Findings of the ACL*, 2024.
- [24] E. J. Hu et al., « LoRA : Low-rank adaptation of large language models », in *International Conference on Learning Representations (ICLR)*, 2022.
- [25] T. Dettmers, A. Pagnoni, A. Holtzman et L. Zettlemoyer, « QLoRA : Efficient finetuning of quantized LLMs », in *NeurIPS*, 2023.
- [26] S. Han, H. Mao et W. J. Dally, « Deep compression : Compressing deep neural networks with pruning, trained quantization and Huffman coding », in *ICLR*, 2016.
- [27] A. E. W. Johnson et al., « MIMIC-IV, a freely accessible electronic health record dataset », *Scientific Data*, vol. 10, 2023.
- [28] A. E. W. Johnson et al., « MIMIC-III, a freely accessible critical care database », *Scientific Data*, vol. 3, 2016.
- [29] E. J. Topol, « High-performance medicine : the convergence of human and artificial intelligence », *Nature Medicine*, vol. 25, p. 44–56, 2019.
- [30] Organisation Mondiale de la Santé, *Ethics and Governance of Artificial Intelligence for Health*. Genève : OMS, 2021.

- [31] D. S. Char, N. H. Shah et D. Magnus, « Implementing machine learning in health care : addressing ethical challenges », *New England Journal of Medicine*, vol. 378, n° 11, p. 981–983, 2018.
- [32] S. Levin et al., « Machine-learning-based electronic triage more accurately differentiates patients with respect to clinical outcomes compared with the Emergency Severity Index », *Annals of Emergency Medicine*, vol. 71, n° 5, p. 565–574, 2018.
- [33] Y. Raita et al., « Emergency department triage prediction of clinical outcomes using machine learning models », *Critical Care*, vol. 23, 2019.
- [34] J.-M. Kwon et al., « Validation of deep-learning-based triage and acuity score using a large national dataset », *PLOS ONE*, vol. 13, n° 10, 2018.
- [35] O. Ivanov et al., « Improving ED emergency severity index acuity assignment using machine learning and clinical natural language processing », *Journal of Emergency Nursing*, vol. 47, n° 2, p. 265–278, 2021.
- [36] K. Singhal et al., « Large language models encode clinical knowledge », *Nature*, vol. 620, p. 172–180, 2023.
- [37] G. Gerganov et al., « llama.cpp : LLM inference in C/C++ », dépôt GitHub, github.com/ggml-org/llama.cpp, 2023.
- [38] A. Paszke et al., « PyTorch : An imperative style, high-performance deep learning library », in *NeurIPS*, 2019.
- [39] T. Wolf et al., « Transformers : State-of-the-art natural language processing », in *EMNLP (System Demonstrations)*, 2020.
- [40] S. Mangrulkar et al., « PEFT : State-of-the-art parameter-efficient fine-tuning methods », dépôt GitHub, github.com/huggingface/peft, 2022.
- [41] PostgreSQL Global Development Group, *PostgreSQL 15 Documentation*, 2025.
- [42] K. Myśliwiec, *NestJS Documentation*, docs.nestjs.com, 2025.
- [43] Meta Platforms, *React Documentation*, react.dev, 2025.
- [44] Z. Chen et al., « MEDITRON-70B : scaling medical pretraining for large language models », *arXiv preprint arXiv :2311.16079*, 2023.
- [45] T. Han et al., « MedAlpaca : an open-source collection of medical conversational AI models and training data », *arXiv preprint arXiv :2304.08247*, 2023.

Exemples de cas du jeu de données

Les deux cas suivants illustrent le format d'un exemple : une entrée patient et la sortie attendue, exprimée dans le schéma de production décrit à l'Annexe B (niveau, couleur, confiance et score sur 100, diagnostics codés en CIM-10, examens, précautions, explication et indicateurs). Construits manuellement, ils ne correspondent à aucun patient réel.

Cas ORANGE : douleur thoracique (CASE-001).

```
{
  "id": "CASE-001",
  "input": {
    "age": 45,
    "gender": "M",
    "chief_complaint": "Douleur thoracique oppressive",
    "symptoms": ["douleur_thoracique", "dyspnee", "sudation", "nausees"],
    "pain_level": 8,
    "pain_locations": ["thorax_centre", "bras_gauche"],
    "vitals": {
      "temperature": 37.0,
      "bp_sys": 165,
      "bp_dia": 95,
      "heart_rate": 110,
      "spo2": 94,
      "respiratory_rate": 22,
      "blood_glucose": null
    },
    "history": ["hypertension", "diabete_type2"],
    "medications": ["metformine", "lisinopril"],
    "allergies": [],
    "arrival_mode": "WALK_IN",
    "onset": "il y a 2 heures",
    "progression": "WORSENING"
  },
  "output": {
    "level": 2,
    "color": "ORANGE",
    "confidence": 85,
    "score": 80,
    "maxWaitMinutes": 10,
    "diagnoses": [
      {"code": "I21.9", "name": "Infarctus du myocarde aigu", "probability": 78}
    ],
    "recommendedTests": ["ECG", "Troponine", "NFS", "Ionogramme"],
  }
}
```

```

    "precautions": ["Position semi-assise", "Monitoring continu", "Voie veineuse"],
    "explanation": "Homme de 45 ans, douleur thoracique oppressive irradiant au bras gauche, sueurs et dyspnée, SpO2 94 %, antécédents d'HTA et de diabète. Tableau évocateur d'un syndrome coronarien : priorité ORANGE.",
    "flags": ["IMMEDIATE_ATTENTION", "SPECIALIST_CARDIOLOGIE"]
  }
}

```

Cas RED : polytraumatisme (TRAU-002).

```

{
  "id": "TRAU-002",
  "input": {
    "age": 45,
    "gender": "M",
    "chief_complaint": "Polytraumatisme - accident autoroute Est-Ouest",
    "symptoms": ["polytraumatisme", "perte_connaissance", "hemorragie_multiple",
      "fracture_bassin", "GCS_8"],
    "pain_level": 10,
    "pain_locations": ["thorax", "bassin", "tete"],
    "vitals": {
      "temperature": 35.9,
      "bp_sys": 80,
      "bp_dia": 45,
      "heart_rate": 135,
      "spo2": 88,
      "respiratory_rate": 32,
      "blood_glucose": null
    },
    "history": ["hypertension"],
    "medications": ["amlodipine"],
    "allergies": [],
    "arrival_mode": "SAMU",
    "onset": "il y a 20 minutes",
    "progression": "WORSENING"
  },
  "output": {
    "level": 1,
    "color": "RED",
    "confidence": 92,
    "score": 95,
    "maxWaitMinutes": 0,
    "diagnoses": [
      {"code": "T07", "name": "Polytraumatisme avec choc hémorragique", "probability": 95},
      {"code": "S32.9", "name": "Fracture du bassin", "probability": 85}
    ],
    "recommendedTests": ["Body scanner", "NFS urgente", "Groupage RAI",

```

```
        "Bilan complet", "Gaz du sang", "Lactates"],  
    "precautions": ["Intubation probable", "Remplissage vasculaire",  
        "Transfusion urgente", "Immobilisation rachis",  
        "Monitoring continu"],  
    "explanation": "Homme de 45 ans, polytraumatisme apres accident, Glasgow 8,  
    hemorragies multiples, TA 80/45 et FC 135 traduisant un choc hemorragique, SpO2  
    88 %. Urgence vitale immediate : priorite ROUGE.",  
    "flags": ["IMMEDIATE_ATTENTION", "TRAUMA_MAJOR", "CHOC_HEMORRAGIQUE"]  
    }  
}
```

Prompt système de triage (extrait)

Extrait du prompt système réel utilisé à l'entraînement et à l'inférence : schéma de sortie JSON, exemple des règles du niveau 1 et règles de sécurité. Les règles des niveaux 2 à 5 sont omises.

You are BetterEMS, an emergency triage AI assistant **for** Algerian hospitals. You follow the Manchester Triage Protocol (MTS) strictly.

Given patient data, respond ONLY with valid JSON in the exact format below. No extra text.

```
{
  "level": <1-5>,
  "color": "<RED|ORANGE|YELLOW|GREEN|BLUE>",
  "confidence": <0-100>,
  "score": <15|35|60|80|95>,
  "maxWaitMinutes": <0|10|60|120|240>,
  "diagnoses": [
    { "code": "<ICD-10 code>", "name": "<diagnosis in French>", "probability":
      <0-100> }
  ],
  "recommendedTests": [ "<test1>", "<test2>" ],
  "precautions": [ "<precaution1>", "<precaution2>" ],
  "explanation": "<Clinical reasoning in French: patient summary, vitals status,
    key Manchester discriminant, probable diagnosis, conclusion>",
  "flags": [ "<FLAG1>", "<FLAG2>" ]
}
```

MANCHESTER TRIAGE RULES (first match = final level):

LEVEL 1 - RED (Immediat, score=95, 0 min):

- Compromised airway
- Inadequate breathing (SpO2 <90%)
- No pulse / unresponsive
- Currently fitting (seizure)
- Severe hypoglycemia (<3 mmol/L)
- Major uncontrolled hemorrhage
- Shock state (BP sys <80, HR >130, altered consciousness)
- Cardiac arrest

[...]

FLAGS:

- IMMEDIATE_ATTENTION: levels 1-2
- SPECIALIST_<NAME>: when specialist needed (CARDIOLOGIE, NEUROLOGIE, CHIRURGIE, etc
-.)
- ISOLATION_REQUIRED: infectious disease risk

SAFETY RULES:

- When confidence <60, add flag REQUIRES_MANUAL_REVIEW and escalate one level up
- When red flags present but logic suggests low urgency, escalate
- ALWAYS err on the side of higher urgency when uncertain

Configuration du fine-tuning (extrait)

Extrait du script d'entraînement (LoRA fp16) : hyperparamètres, configuration LoRA et arguments d'entraînement.

```
# LoRA - smaller rank since no quantization = better base weights
LORA_R = 32
LORA_ALPHA = 64
LORA_DROPOUT = 0.05

# Training config
NUM_EPOCHS = 3
BATCH_SIZE = 1          # Smaller batch to fit the fp16 model on a single GPU
GRADIENT_ACCUMULATION = 16 # Effective batch = 16
LEARNING_RATE = 2e-5
MAX_SEQ_LENGTH = 2048    # Sized to the average output length (~800 chars)
WARMUP_RATIO = 0.05
WEIGHT_DECAY = 0.01

# [...]

lora_config = LoraConfig(
    r=LORA_R,
    lora_alpha=LORA_ALPHA,
    lora_dropout=LORA_DROPOUT,
    bias="none",
    task_type="CAUSAL_LM",
    target_modules=[
        "q_proj", "k_proj", "v_proj", "o_proj",
        "gate_proj", "up_proj", "down_proj",
    ],
)

# [...]

training_args = SFTConfig(
    num_train_epochs=NUM_EPOCHS,
    per_device_train_batch_size=BATCH_SIZE,
    gradient_accumulation_steps=GRADIENT_ACCUMULATION,
    learning_rate=LEARNING_RATE,
    weight_decay=WEIGHT_DECAY,
    warmup_ratio=WARMUP_RATIO,
    lr_scheduler_type="cosine",
    load_best_model_at_end=True,
```

```
metric_for_best_model="eval_loss",  
fp16=True,           # fp16 mixed precision (NOT bf16)  
gradient_checkpointing=True,  
max_grad_norm=1.0,   # Standard gradient clipping  
optim="adamw_torch", # Standard optimizer (not paged)  
# [...]  
)
```

Intégration du modèle dans le backend (extrait)

Extrait du service `ai.service.ts` (NestJS) : appel du serveur d'inférence via l'API compatible OpenAI, puis bascule vers l'algorithme de secours en cas d'échec. L'adresse du serveur est lue en configuration (valeur générique ici : `http://serveur-ia:8080`).

```
async callOllama(prompt: string): Promise<string> {
  const config = await this.getAiConfig();

  // Try OpenAI-compatible API first (llama-server)
  try {
    const response = await fetch(`${config.aiUrl}/v1/chat/completions`, {
      method: 'POST',
      headers,
      body: JSON.stringify({
        model: config.model,
        messages: [
          { role: 'system', content: MANCHESTER_TRIAGE_SYSTEM_PROMPT },
          { role: 'user', content: prompt },
        ],
        temperature: 0.1,
        max_tokens: 2048,
      }),
    });

    if (response.ok) {
      const data = (await response.json()) as {
        choices: Array<{ message: { content: string } }>;
      };
      return data.choices[0].message.content;
    }
  } catch (error) {
    this.logger.warn(`OpenAI API not available: ${error}`);
  }
  // [...]
}

async triage(visitData: VisitDataForTriage): Promise<TriageAiResult> {
  const prompt = this.buildPrompt(visitData);
  try {
    const rawResponse = await this.callOllama(prompt);
```

```
const result = this.parseTriageResponse(rawResponse);  
// Garde-fou de confiance : sous le seuil, on plafonne l'urgence à ORANGE  
// (un cas déjà ROUGE/ORANGE reste inchangé) et on marque le cas pour revue  
if (result.confidence < 50) {  
  result.flags.push('LOW_CONFIDENCE');  
  if (result.level > TriageLevel.ORANGE) {  
    result.level = TriageLevel.ORANGE;  
    result.color = 'ORANGE';  
    result.maxWaitMinutes = 10;  
    result.flags.push('ESCALATED_DUE_TO_LOW_CONFIDENCE');  
  }  
}  
return result;  
} catch (error) {  
  this.logger.warn(`AI triage failed, using algorithmic fallback: ${error}`);  
  const algoResult = this.generateAlgorithmicTriage(visitData);  
  algoResult.flags.push('AI_SERVICE_ERROR');  
  return algoResult;  
}  
}
```

Configuration de déploiement (extrait)

Extrait du fichier docker-compose.yml utilisé pour les services de données (mots de passe masqués).

```
services:
  postgres:
    image: postgres:16-alpine
    container_name: betterems-db
    environment:
      POSTGRES_DB: betterems
      POSTGRES_USER: betterems
      POSTGRES_PASSWORD: "*****"
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U betterems"]
      interval: 5s
      timeout: 5s
      retries: 5

  redis:
    image: redis:7-alpine
    container_name: betterems-redis
    ports:
      - "6379:6379"
    volumes:
      - redisdata:/data

volumes:
  pgdata:
  redisdata:
```

Captures d'écran de l'application

Cette annexe présente les principales interfaces de BetterEMS en fonctionnement, dans l'ordre du parcours patient. Sur certaines captures, le moteur d'inférence apparaît sous l'étiquette interne de déploiement « BetterEMS-Triage-v2 » : il s'agit du même modèle que celui décrit dans le mémoire, le suffixe désignant la version mise en production.

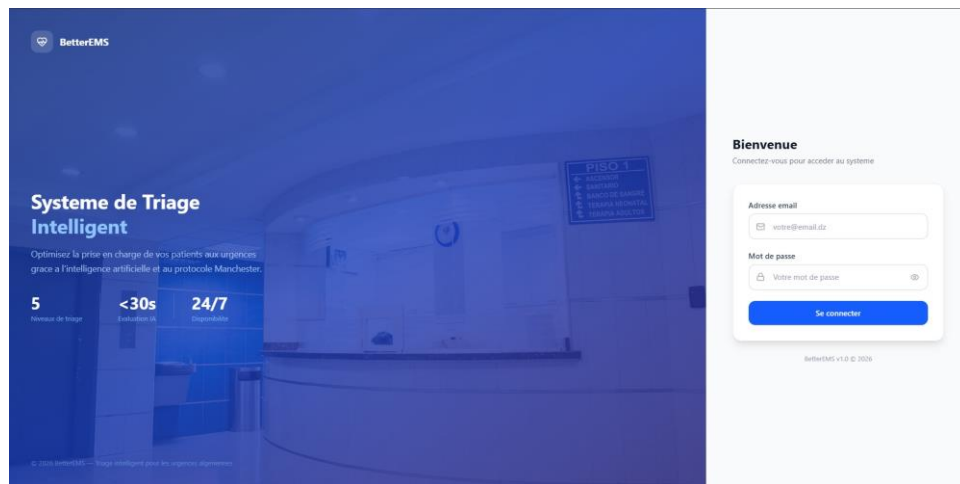


Fig. F.1 : Page de connexion : authentification du personnel par rôle.

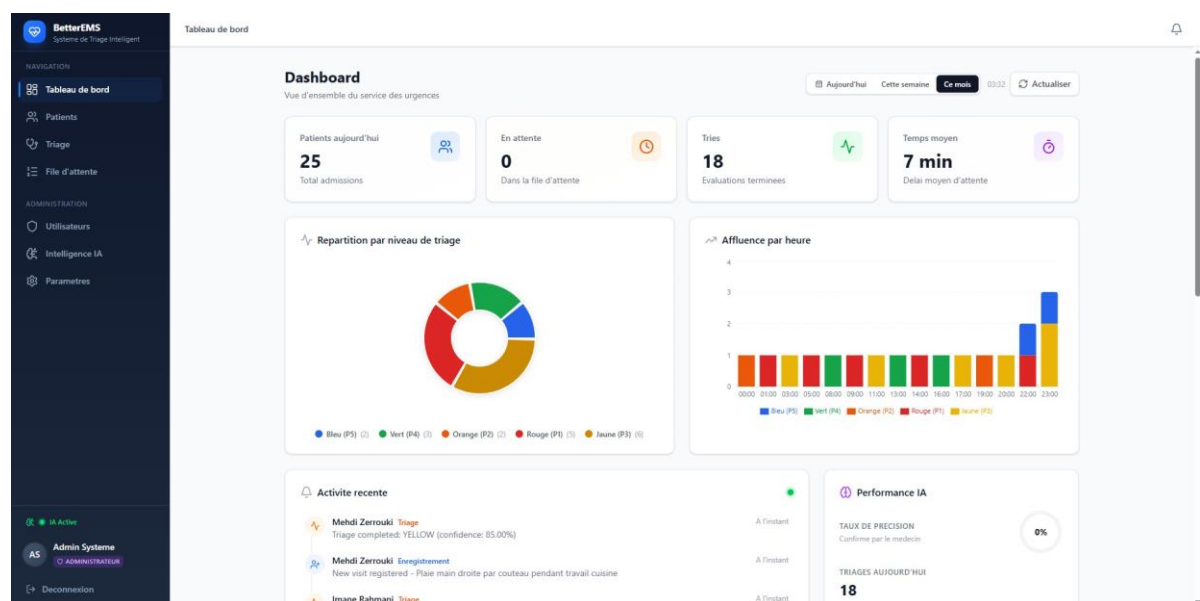


Fig. F.2 : Tableau de bord : vue d'ensemble du service (volumes, niveaux de gravité, temps d'attente).

The screenshot shows the 'Identite du patient' form in the BetterEMS application. The form is titled 'Identite du patient' and includes a note: 'Tous les champs sont optionnels pour les cas urgents.' The form fields are as follows:

Identite	
Nom	Prénom
BENALI	Mohamed
Age	Sexe
45 ans	Homme
N° Telephone	N° Identite
+213 555 123 456	Optionnel
N° Sécurité sociale	
Optionnel	

Buttons: 'Retour' (top left), 'Suivant' (bottom right).

Fig. F.3 : Enregistrement d'un patient par l'agent d'accueil (étape 1 sur 5).

The screenshot shows the 'Signes vitaux' form in the BetterEMS application. The form is titled 'Signes vitaux' and includes a note: 'Entrez les valeurs mesurées. Les champs hors normes s'affichent en rouge.' The form fields are as follows:

Signes vitaux	
Température	Fréquence cardiaque
37.0 °C	80 bpm
Pression artérielle	Saturation O ₂
120 / 80 mmHg	98 %
Fréquence respiratoire	Glycémie
16 /min	100 mg/dL
Score de Glasgow	
15 /15	
Niveau de douleur global	
3/10	

Buttons: 'Précédent' (top left), 'Suivant' (bottom right).

Fig. F.4 : Enregistrement d'un patient (étape 2 sur 5).

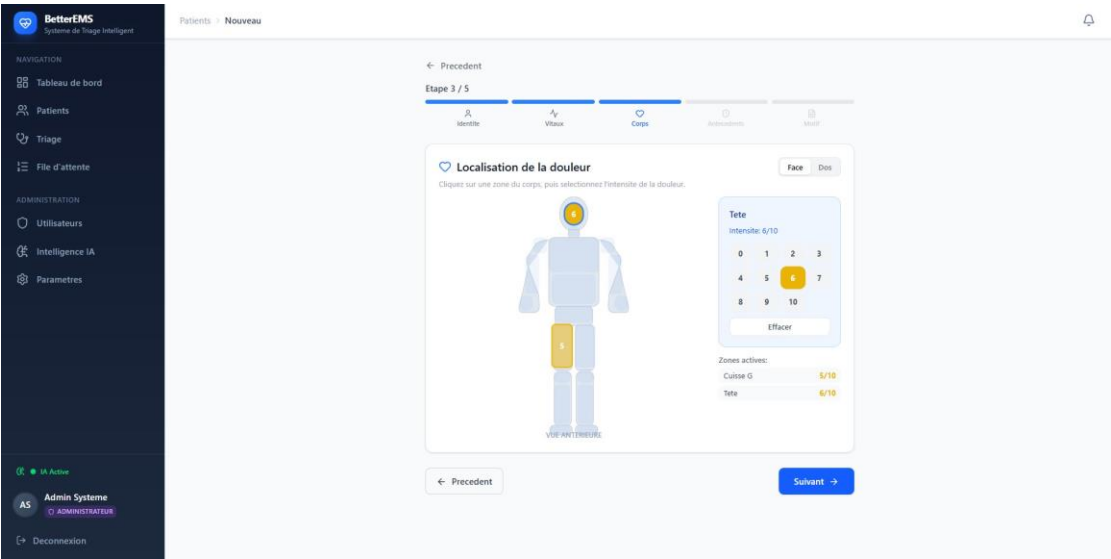


Fig. F.5 : Enregistrement d'un patient (étape 3 sur 5).

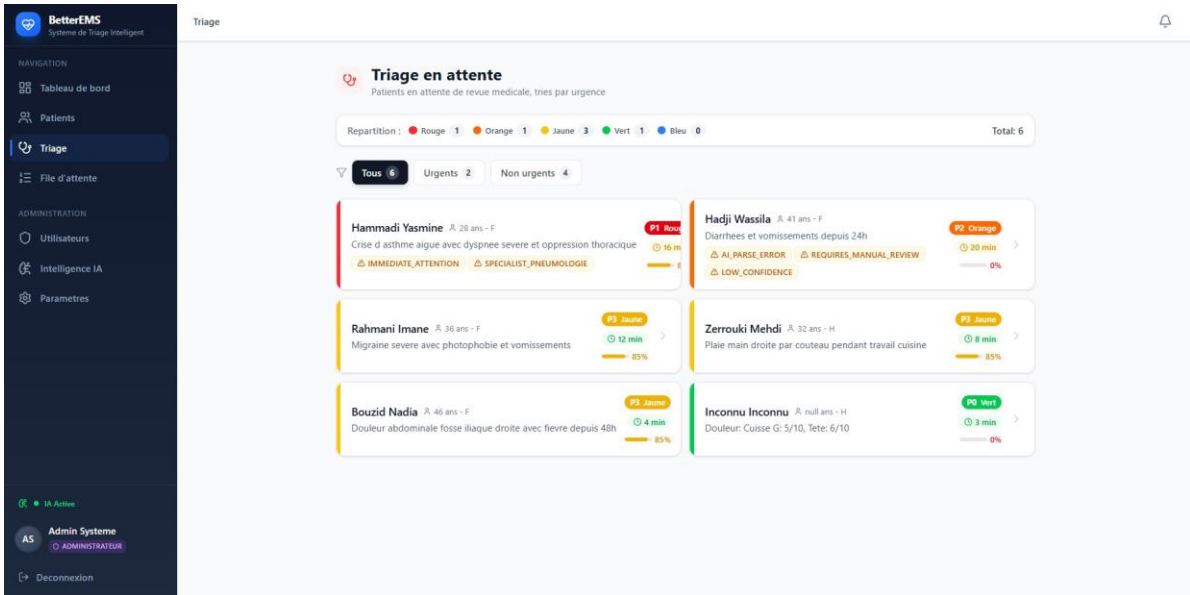


Fig. F.6 : Écran de triage : patients triés par niveau d'urgence, avec couleur Manchester, score de confiance et indicateurs d'alerte produits par le modèle.

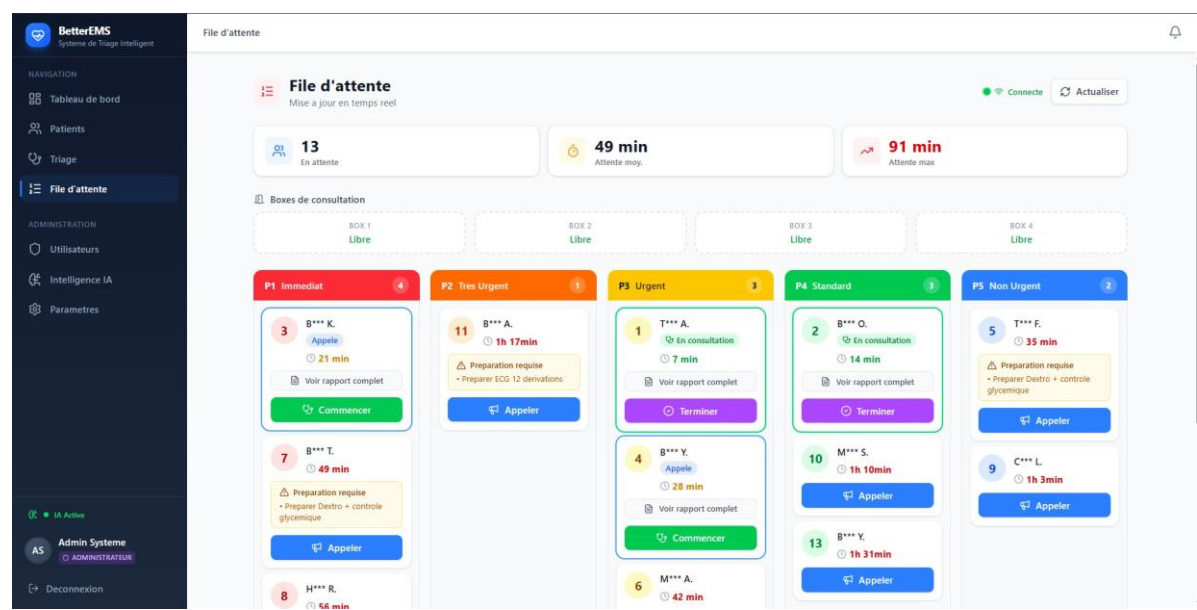


Fig. F.7 : File d’attente priorisée, mise à jour en temps réel.

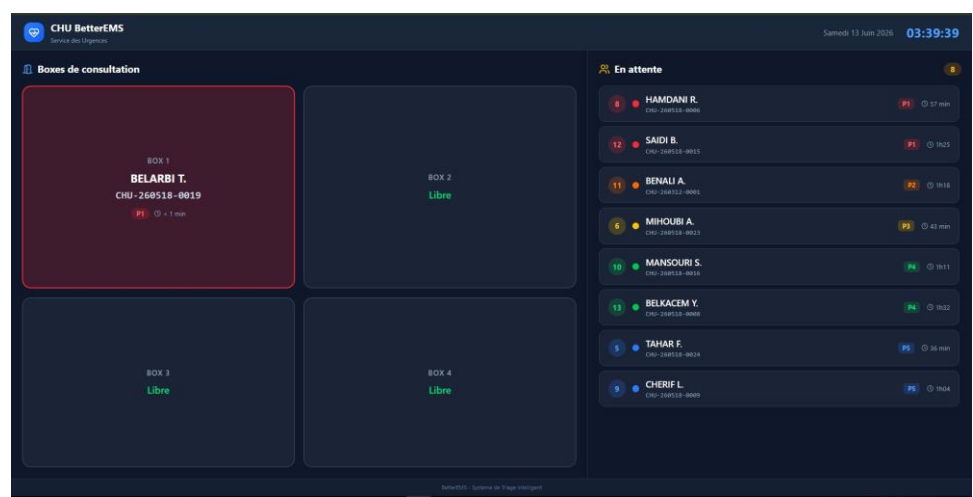


Fig. F.8 : Écran d’appel de la salle d’attente (affichage TV) : numéro de ticket, box et couleur du patient appelé.

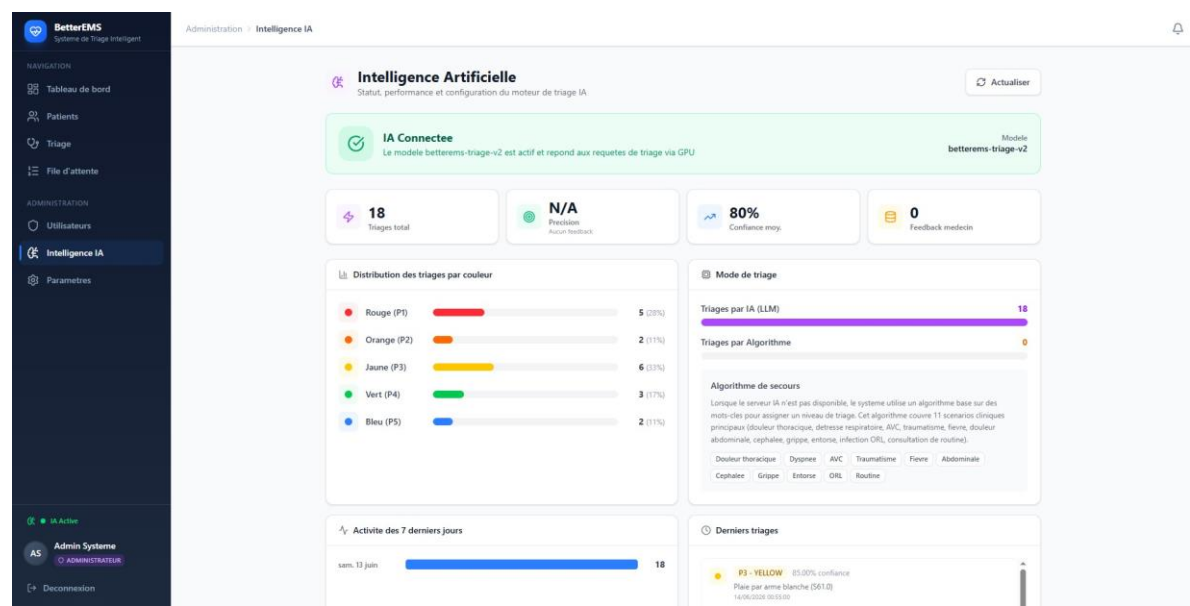


Fig. F.9 : Console d'administration du moteur d'intelligence artificielle (état du service, disponibilité, repli).

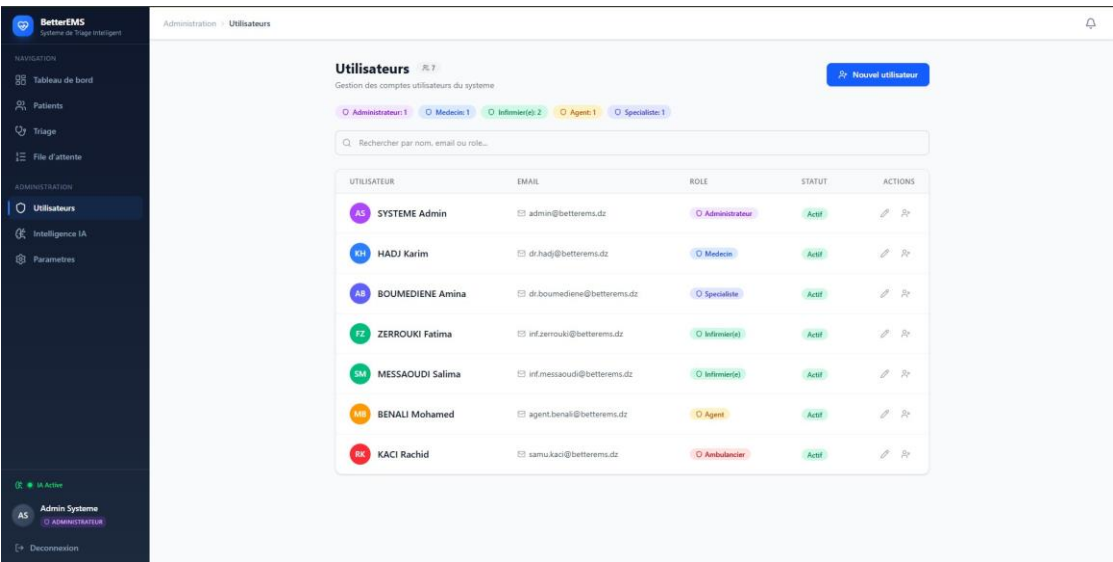


Fig. F.10 : Gestion des utilisateurs et des rôles (administrateur).

Glossaire

Triage : classement des patients d'un service d'urgences par ordre de gravité clinique, indépendamment de leur ordre d'arrivée.

MTS (Manchester Triage System) : protocole de triage à cinq niveaux (Rouge, Orange, Jaune, Vert, Bleu) associant à chaque niveau un délai maximal de prise en charge.

Discriminateur : critère clinique précis (par exemple $SpO_2 < 90\%$) qui, dans le protocole de Manchester, détermine le niveau de triage d'un patient.

LLM (Large Language Model) : modèle de langage de grande taille entraîné sur de vastes corpus de textes et capable de comprendre et de produire du langage naturel.

Token : unité élémentaire de texte (mot, fragment de mot ou symbole) manipulée par un modèle de langage.

Fine-tuning : ré-entraînement d'un modèle pré-entraîné sur un jeu de données spécialisé afin de l'adapter à une tâche précise.

LoRA (Low-Rank Adaptation) : technique de fine-tuning qui gèle les poids du modèle et n'entraîne que de petites matrices basse-rang ajoutées en parallèle, réduisant fortement le coût d'entraînement.

Rang (rank) : dimension des matrices basse-rang de LoRA ; plus le rang est élevé, plus l'adaptation a de capacité mais plus elle coûte en mémoire.

Quantification : réduction de la précision numérique des poids d'un modèle (par exemple de 16 à 4 bits) pour diminuer sa taille et sa consommation mémoire.

GGUF : format de fichier compact destiné à l'exécution de modèles de langage avec llama.cpp, y compris en versions quantifiées.

Inférence : phase d'utilisation d'un modèle entraîné pour produire des prédictions sur de nouvelles données.

VRAM : mémoire embarquée sur la carte graphique (GPU), qui limite la taille des modèles chargeables et entraînaibles.

Prompt système : instructions permanentes fournies au modèle de langage avant chaque requête pour définir son rôle, ses règles et son format de sortie.

JSON : format texte structuré d'échange de données, utilisé ici comme format de sortie du modèle et de communication entre les composants du système.

API REST : interface de communication entre applications fondée sur le protocole HTTP et des ressources identifiées par des URL.

WebSocket : protocole de communication bidirectionnelle persistante entre navigateur et serveur, utilisé pour la mise à jour en temps réel de la file d'attente.

Fallback : mécanisme de secours qui prend automatiquement le relais quand le composant principal est indisponible (ici, un algorithme à règles remplace le modèle).

Human-in-the-loop : principe selon lequel un humain (ici le médecin) valide ou corrige chaque décision de l'IA avant qu'elle ne soit appliquée.

RBAC (Role-Based Access Control) : contrôle d'accès attribuant à chaque utilisateur des permissions selon son rôle (agent, infirmier, médecin, administrateur).

Surapprentissage : situation où un modèle mémorise les données d'entraînement au lieu de généraliser, ce qui dégrade ses performances sur des données nouvelles.

Résumé

Les services d'urgences hospitalières fonctionnent souvent au-delà de leur capacité, et l'ordre d'arrivée, qui reste la règle de fait dans de nombreux établissements algériens, ne distingue pas un cas bénin d'une urgence vitale. Ce mémoire présente BetterEMS, un système d'aide au triage qui priorise les patients selon le protocole de Manchester à l'aide d'un modèle de langage médical spécialisé. Nous avons constitué un corpus propriétaire de cas de triage en français, adapté au contexte algérien, puis adapté le modèle BioMistral-7B par la méthode LoRA pour obtenir notre modèle BetterEMS-Triage, qui atteint environ 96 % de précision au niveau du token sur l'ensemble de validation et rend sa proposition en deux à quatre secondes par patient. Le modèle est intégré dans un système web complet, temps réel, où le médecin valide ou corrige chaque décision. Le travail a par ailleurs été valorisé sous la forme d'une startup labellisée par l'incubateur de l'Université de Saïda.

Mots-clés : triage médical, protocole de Manchester, modèle de langage, fine-tuning LoRA, BioMistral, files d'attente prioritaires, urgences hospitalières.

Abstract. Hospital emergency departments routinely operate beyond capacity, and first-come-first-served ordering, still the de facto rule in many Algerian hospitals, does not distinguish a minor complaint from a life-threatening emergency. This thesis presents BetterEMS, a triage decision-support system that prioritises patients according to the Manchester protocol using a specialised medical language model. We built a proprietary French-language triage corpus tailored to the Algerian setting and adapted the BioMistral-7B model through LoRA fine-tuning, yielding our BetterEMS-Triage model, which reaches about 96% token-level accuracy on the validation set and returns a recommendation within two to four seconds per patient. The model is embedded in a complete real-time web system in which the physician validates or corrects every decision. The project has also been spun off as a startup certified by the University of Saïda incubator.

Keywords: medical triage, Manchester protocol, language model, LoRA fine-tuning, BioMistral, priority queues, emergency department

المخلص. تعمل مصالح الاستعجالات الاستشفائية غالبًا بما يتجاوز طاقتها، وترتيب المرضى حسب وقت الوصول لا يميز بين الحالة البسيطة والحالة التي تهدد الحياة. تعرض هذه المذكرة نظام BetterEMS، وهو نظام للمساعدة على الفرز الطبي يرتب المرضى وفق بروتوكول مانشستر باستعمال نموذج لغوي طبي متخصص. قمنا ببناء مدونة خاصة لحالات الفرز باللغة الفرنسية مكيفة مع السياق الجزائري، ثم كیفنا النموذج BioMistral-7B بطريقة LoRA للحصول على نموذجنا BetterEMS-Triage الذي يبلغ دقة تقارب ٩٦% على مجموعة التحقق ويُصدر اقتراحه في ظرف ثانيتين إلى أربع ثوانٍ لكل مريض. النموذج مدمج في نظام ويب كامل يعمل في الزمن الحقيقي، حيث يصادق الطبيب على كل قرار أو يصحّحه. كما تُمن هذا العمل في شكل مؤسسة ناشئة معتمدة من حاضنة جامعة سعيدة.

الكلمات المفتاحية: الفرز الطبي، بروتوكول مانشستر، النماذج اللغوية، الضبط الدقيق LoRA، BioMistral، طوابير الأولوية، مصالح الاستعجالات.