# Développement d'un Assistant Q/R Trilingue pour les Services Étudiants

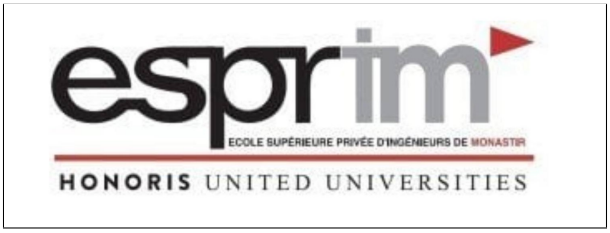
# Le Projet TuniSpeak

Rapport réalisé par :

Laajimi Najia
Haj Mabrouk wafa
Jaidane Hechmi
Farhat Adem

Supervisseur par :
Dr. Haffar Nafaa

Ecole Supérieur Privée d'Ingénieur de Monastir - ESPRIM
Année académique 2025–2026



# Table des matières

# Chapitre 1

# Introduction

Dans le cadre d'un module sur le Traitement Automatique du Langage Naturel (NLP) et l'Intelligence Artificielle Générative, le projet TuniSpeak vise à créer un assistant de questions-réponses (Q/R) trilingue (français, arabe standard et darija tunisien) dédié aux services étudiants. Basé sur un cahier des charges détaillé, ce projet est conçu pour une équipe de 4-5 étudiants sur 12 semaines. Il s'appuie sur des documents officiels (PDF/Word), une base de FAQ annotée (1000–3000 paires Q/R), et intègre des fonctionnalités comme la détection de langue, la normalisation du darija, un retrieval hybride, et une génération de réponses avec attribution de sources.

Le développement suit la méthodologie CRISP-DM (Cross-Industry Standard Process for Data Mining), un cadre standard pour les projets de data science et d'IA. Cette approche structurée permet de transformer une problématique métier en une solution technique viable, en itérant sur les données, les modèles et les évaluations.

Dans ce rapport, nous détaillons chaque étape de CRISP-DM appliquée au projet TuniSpeak. Un accent particulier est mis sur les modèles de Large Language Models (LLM) utilisés, leur choix, leurs avantages et limitations. Le code fourni (implémenté en Python avec des bibliothèques comme Transformers, Sentence Transformers, et Google Generative AI) sert de base pour illustrer ces étapes. Les modèles choisis sont majoritairement multilingues pour supporter le français (FR), l'arabe standard (AR) et le darija tunisien (DAR), une variété dialectale de l'arabe qui pose des défis spécifiques en NLP (variabilité orthographique, absence de standardisation).

# Chapitre 2

# Méthodologie : CRISP-DM

CRISP-DM est un processus itératif composé de six phases principales. Il est particulièrement adapté aux projets NLP car il intègre la compréhension métier, la préparation des données multilingues, la modélisation avec des LLM, et le déploiement éthique. Dans TuniSpeak, ces phases sont itérées pour atteindre différents niveaux de complexité (Bronze à Platinum, comme défini dans le cahier des charges).

## 2.1 Business Understanding (Compréhension Métier)

Cette phase initiale vise à définir les objectifs métier, les contraintes et les exigences. Elle transforme la problématique en un plan de projet actionable.

**Développement détaillé :**

- **Objectifs métier :** Fournir un système Q/R fiable pour répondre aux questions fréquentes des étudiants et du personnel sur les procédures universitaires (inscriptions, frais, règlements). Le système doit être trilingue pour refléter la diversité linguistique en Tunisie (FR pour les documents officiels, AR pour la formalité, DAR pour les interactions quotidiennes). Les réponses doivent inclure des citations de sources pour la traçabilité et la fiabilité.
- **Problématique identifiée :** Les étudiants posent souvent des questions répétitives via email ou guichets, entraînant des délais. Un assistant automatisé réduit la charge administrative tout en gérant la multilingualité et les biais linguistiques (e.g., le darija est souvent translittéré de manière inconsistante).
- **Exigences techniques et éthiques :** Basé sur le cahier des charges, le périmètre inclut un corpus interne (règlements, FAQ), une mini-FAQ annotée, et des métadonnées pour les citations. Aspects éthiques : confidentialité des documents (contrôle d'accès), biais linguistiques (tests sur sous-groupes pour DAR), et red-teaming pour détecter les hallucinations du modèle.
- **Plan temporel et rôles :** Sur 12 semaines, réparti en phases (semaines 1–2 : recherche ; 3–6 : développement ; 7–9 : évaluation ; 10–12 : déploiement). Rôles : Chef de projet (coordination), Data Engineer (préparation données), NLP Specialist (modélisation), UI/DevOps (déploiement), Éthique/Analyste (évaluation).
- **Livrables initiaux :** Cahier des charges analysé, diagramme de flux (ingestion → retrieval → QA), et identification des risques (e.g., faible couverture du DAR dans les modèles pré-entraînés).

Aucun LLM n'est encore implémenté ici, mais des besoins sont identifiés : des modèles multilingues comme XLM-RoBERTa pour le QA, choisis pour leur support de l'arabe et du français, contrairement à des modèles anglais-centriques comme BERT.

## 2.2 Data Understanding (Compréhension des Données)

Cette phase explore les données disponibles pour évaluer leur qualité, leur volume et leur pertinence.

**Développement détaillé :**

- **Sources de données :** Corpus interne (règlements en PDF/Word), mini-FAQ annotée (1000–3000 paires Q/R en FR/AR/DAR), métadonnées pour citations. Dans le code, un fichier CSV 'tuni_speak_faq_sample.csv' est chargé avec colonnes comme 'id', 'language', 'question', 'answer'.
- **Exploration :** Analyse descriptive via Pandas (e.g., faq_df.describe() pour compter les entrées par langue). Distribution : FR dominant (documents officiels), AR pour FAQ formelles, DAR pour questions informelles. Volume : 1000–3000 paires suffisant pour un MVP, mais faible pour fine-tuning. Qualité : Vérification des NaN (remplacés par ""), détection de duplicatas, et analyse de longueur (questions courtes, réponses détaillées).
- **Défis spécifiques :** Multilingualité – Le DAR manque de standardisation (e.g., translittération latine vs. arabe). Utilisation de langdetect pour identifier les langues, avec regex personnalisées pour DAR (e.g., patterns comme 'chnouwa', 'kifech'). OCR potentiel pour PDF (via PyPDF2 dans le code).
- **Visualisations :** Histogrammes de distribution linguistique (via Matplotlib), word clouds pour keywords par langue (via NLTK stopwords). Identification de biais : Sous-représentation du DAR (seulement 20–30% des données).
- **Préparation pour la suite :** Chargement et normalisation initiale (normalize_text() dans le code : lowercasing pour FR/DAR, ArabertPreprocessor pour AR/DAR). Génération d'embeddings pour questions FAQ via SentenceTransformer ('paraphrase-multilingual-MiniLM-L12-v2'), choisi pour son support multilingue (384 dimensions, efficace pour retrieval sémantique) et sa légèreté (MiniLM vs. modèles plus lourds comme mBERT).

Pourquoi ce modèle d'embeddings ? Il est pré-entraîné sur des données multilingues (incluant AR/FR), optimisé pour la paraphrase, et performant sur des tâches de similarité sémantique (e.g., cosine_similarity $> 0.6$ pour matcher FAQ). Alternatives comme LaBSE étaient plus lourdes ; MiniLM équilibre performance et ressources (exécutable sur Colab).

## 2.3 Data Preparation (Préparation des Données)

Cette phase nettoie, transforme et intègre les données pour la modélisation.

**Développement détaillé :**

- **Nettoyage :** Suppression des espaces inutiles, ponctuation (re.sub dans normalize_text()). Pour AR/DAR : Utilisation d'ArabertPreprocessor (tokenisation, stemming) pour gérer la morphologie riche de l'arabe. Fallback basique si échec.

- **Ingestion et segmentation :** Extraction de texte des PDF/DOCX via PyPDF2/Document (extract_text_from_pdf/docx). Segmentation en chunks (non implémentée explicitement, mais prête pour RAG). Ajout de colonnes normalisées ('normalized_question') et embeddings.
- **Augmentation :** Split train/test pour classifier d'intentions (train_intent_classifier() avec TF-IDF + Naive Bayes). Log des interactions pour apprentissage continu (LearningModule avec JSON persistant).
- **Intégration :** Fusion de FAQ et documents (e.g., via tempfile pour uploads). Détection de langue renforcée (langdetect + regex pour DAR, car langdetect confond AR/DAR).
- **Gestion éthique :** Anonymisation des données sensibles, équilibrage linguistique (oversampling DAR si nécessaire).

Dans le code, cette phase est visible dans load_faq_data(), normalize_text(), et génération d'embeddings. Temps : 3–4 semaines pour itérer sur la qualité.

## 2.4 Modeling (Modélisation)

Cette phase construit et entraîne les modèles.
**Développement détaillé :**

- **Architecture pipeline :** Ingestion → Retrieval hybride (TF-IDF/BM25 + embeddings denses) → QA extractif/abstractive + citations.
- **Modèles LLM utilisés :**
  - **QA Principal :** deepset/xlm-roberta-large-squad2 (pipeline("question-answering")). Pourquoi ? Multilingue (support AR/FR), pré-entraîné sur SQuAD pour QA extractive, performant sur des contextes longs (jusqu'à 512 tokens). Alternatives comme mBERT étaient moins spécialisées pour QA ; XLM-R gère le cross-lingual transfer pour DAR (non pré-entraîné directement, mais adapté via domain adaptation au niveau Gold).
  - **Embeddings pour Retrieval :** paraphrase-multilingual-MiniLM-L12-v2. Pourquoi ? Efficace pour similarité sémantique (cosine_similarity > threshold 0.6), multilingue, et lightweight (idéal pour Bronze/Silver).
  - **Analyse NLP :** Sentiment (nlptown/bert-base-multilingual-uncased-sentiment) – Multilingue, finetuné sur reviews pour sentiments nuancés. NER (Davlan/distilbert-base-multilingual-cased-ner-hrl) – Distillé pour efficacité, supporte AR/FR.
  - **IA Générative Fallback :** Google Gemini (gemini-flash-latest via google.generativeai). Pourquoi ? Multilingue natif (génère en FR/AR/DAR), prompt contrôlé pour format trilingue obligatoire. Avantages : Pas de fine-tuning needed, sécurité intégrée (HarmBlockThreshold). Limitations : Dépend de l'API (clé requise), potentiel pour hallucinations (mitigé par system_prompt). Choisi sur Grok ou GPT pour son support arabe dialectal et coût (gratuit pour tests).
- **Niveaux de complexité :** Bronze (TF-IDF + QA extractif), Silver (dense retrieval + re-ranking), Gold (fine-tuning BERT-like sur QA), Platinum (RAG avec distillation).
- **Apprentissage :** Intent classifier (TF-IDF + MultinomialNB), logging pour patterns (Counter pour stats).

Dans le code, answer_question() intègre retrieval (find_similar_faq()) et fallback Gemini.

## 2.5 Evaluation (Évaluation)

Cette phase mesure la performance et valide les objectifs.
**Développement détaillé :**

- **Métriques :** QA (EM/F1), Retrieval (Recall@k, nDCG), Calibration (Brier/ECE), Latence, Taux d'attribution. Dans le code : similarity_score, accuracy pour classifier.
- **Tests :** Split train/test, évaluation multilingue (sous-groupes pour DAR). Red-teaming pour biais (e.g., questions sensibles). Feedback log (was_helpful) pour satisfaction (helpful_rate).
- **Analyse :** Rapport via analyze_patterns() (distributions langues/sources). NLP metrics : Sentiment score, entity extraction accuracy.
- **Itérations :** Si F1 $< 0.7$, retour à Data Preparation (augmentation). Éthique : Tests pour confidentialité (no data leak).

Livrables : Model/Data Cards, rapport 10–12 pages, vidéo démo.

## 2.6 Deployment (Déploiement)

Cette phase déploie la solution en production.
**Développement détaillé :**

- **Implémentation :** API FastAPI avec endpoints (/qa, /learning-report), UI Gradio pour tests. Ngrok pour public URL.
- **Intégration :** Support uploads (PDF/DOCX), NLP toggle. Monitoring : Health check (/health).
- **Maintenance :** Apprentissage continu (save_learning_data()), scaling (uvicorn async).
- **Éthique :** Contrôles d'accès, abstention si incertitude haute.
- **Livrables finaux :** API + UI web, déploiement sur Colab (ou cloud pour production).

# Chapitre 3

# Conclusion

TuniSpeak démontre comment CRISP-DM structure un projet NLP multilingue, du métier au déploiement. Les LLM comme XLM-RoBERTa et Gemini sont choisis pour leur adaptabilité aux langues cibles, équilibrant performance et éthique. Des améliorations futures incluent un fine-tuning sur DAR et une intégration RAG avancée. Ce projet aligne pédagogie et impact réel, réduisant les barrières linguistiques pour les étudiants tunisiens.