ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR
Membre de HONORIS UNITED UNIVERSITIES

Université Al Akhawayn à Ifrane

Mémoire de fin d'études

Titre du projet de fin d'études

Présenté par : Rayane Mestari
Programme : Computer Science – AI & Data Analytics

Encadrant : Dr. Mohammed Mestari

Année universitaire 2024–2025

Composition du jury

- Dr. Mohammed Mestari – Encadrant académique
- Dr. Abdellatif Marghich – Membre du jury
- Dr. Imane Nizar – Membre du jury

Remerciements

Je tiens à remercier mon encadrant, les membres du jury ainsi que toute l'équipe pédagogique pour leur soutien précieux.

Résumé

Ce mémoire présente...

Mots-clés : intelligence artificielle, apprentissage automatique, AUI, data analytics

Abstract

This thesis presents...

Keywords: artificial intelligence, machine learning, AUI, data analytics

Table des matières

Composition du jury **1**

Remerciements **2**

Résumé **3**

Abstract **4**

1 Introduction **6**

1.1 Tokenisation WordPiece en C++ 4

2 Contexte et Problématique **6**

3 Méthodologie **7**

4 Résultats **8**

5 Conclusion et Perspectives **9**

A Annexe A : Informations complémentaires **10**

Chapitre 1

Introduction

Fondements Mathématiques des Techniques de NLP

1. Tokenization

La tokenization segmente un texte T en unités appelées *tokens* $\{t_1, t_2, ..., t_n\}$. C'est une fonction de transformation :

$$f : T \to \{t_i\} \subseteq \Sigma^*$$

avec Σ l'alphabet. Les techniques utilisées incluent :

- Expressions régulières
- Automates finis
- Algorithmes statistiques (ex. pour les langues non segmentées)

2. Stemming et Lemmatization

Stemming

Réduction heuristique basée sur des règles (suffix stripping) :

$$\text{stem}(w) = w' \quad w' \approx \text{racine de } w$$

Exemple : Algorithmes de Porter, Snowball.

Lemmatization

Forme correcte linguistiquement, dépendant de l'analyse morphosyntaxique :

$$\text{lemma}(w, \text{POS}) = l \quad l \in \text{dictionnaire}$$

Basée sur règles grammaticales et dictionnaires.

3. Suppression des Stopwords

Filtrage booléen sur les tokens :

$$f(t) = \begin{cases} 0 & \text{si } t \in \text{stopwords} \\ 1 & \text{sinon} \end{cases}$$

Réduction manuelle de la dimensionnalité de la représentation.

4. Bag-of-Words (BoW)

Modélisation vectorielle : chaque document devient un vecteur creux :

$$\vec{d} = (c_1, c_2, ..., c_n) \quad c_i = \text{nb de } w_i \text{ dans } d$$

Matrice du corpus : $D \in \mathbb{N}^{m \times n}$, avec m documents et n mots.

5. TF-IDF

Formule :

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \log\left(\frac{N}{DF(t)}\right)$$

- $\text{TF}(t, d)$: fréquence du terme dans le document d
- $DF(t)$: nombre de documents contenant t
- N : nombre total de documents

Pondère l'importance des mots rares et spécifiques.

6. Word Embeddings

6.1 Word2Vec

Deux variantes :

- CBOW : prédit un mot à partir de son contexte
- Skip-gram : prédit le contexte à partir d'un mot

Objectif d'apprentissage :

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-k \leq j \leq k, j \neq 0} \log P(w_{t+j}|w_t)$$

$$P(w_O|w_I) = \frac{\exp(v_{w_O}^T v_{w_I})}{\sum_{w \in V} \exp(v_w^T v_{w_I})}$$

Utilise la technique du *negative sampling* pour une approximation efficace.

6.2 GloVe

Basé sur les cooccurrences de mots :

$$w_i^T w_j + b_i + b_j = \log(X_{ij})$$

Fonction de coût :

$$J = \sum_{i,j} f(X_{ij}) \left(w_i^T w_j + b_i + b_j - \log(X_{ij}) \right)^2$$

Optimisé par moindres carrés pondérés.

6.3 FastText

Utilise les n-grammes de caractères :

$$\vec{w} = \sum_{g \in G(w)} z_g$$

- $G(w)$: ensemble de n-grammes du mot w
- z_g : vecteur associé à chaque n-gramme

Robuste aux mots rares et aux variantes morphologiques.

7. Tableau récapitulatif

Méthode	**Type**	**Représentation**	**Mathématiques**
Tokenization	Symbolique	Séquences	FSA, regex
Stemming	Heuristique	Racine approximée	Substitutions
Lemmatization	Linguistique	Forme canonique	Graphe, dictionnaire
Stopwords	Réduction	Filtres booléens	Indicateur logique
BoW	Statistique	Vecteurs creux	Comptage, matrice $\mathbb{N}^{m \times n}$
TF-IDF	Statistique	Poids pondérés	Log, fréquence inverse
Word2Vec	Neuronal	Vecteurs denses	Skip-gram / CBOW + soft
GloVe	Factorisation	Cooccurrence	Moindres carrés pondérés
FastText	Neuronal morphologique	N-grammes de caractères	Moyenne vectorielle

1.1 Tokenisation WordPiece en C++

```cpp
#include <iostream>
using namespace std;

// Fonction de comparaison de cha nes
bool compare(const char* a, const char* b) {
    int i = 0;
    while (a[i] != '\0' && b[i] != '\0') {
        if (a[i] != b[i]) return false;
        i++;
    }
    return a[i] == '\0' && b[i] == '\0';
}

// Fonction pour copier une sous-cha ne
void substring(const char* word, int start, int end, char* result,
   bool prefix) {
    int k = 0;
    if (prefix) {
        result[k++] = '#';
        result[k++] = '#';
    }
    for (int i = start; i < end; i++) {
        result[k++] = word[i];
    }
    result[k] = '\0';
}

// Simule un vocabulaire
const int VOCAB_SIZE = 5;
const char* vocab[VOCAB_SIZE] = {
    "extra", "##ordin", "##aire", "bon", "##jour"
};

// V rifie si le token est dans le vocabulaire
bool in_vocab(const char* token) {
    for (int i = 0; i < VOCAB_SIZE; i++) {
        if (compare(token, vocab[i])) return true;
    }
    return false;
```

```cpp
}

// Tokeniseur WordPiece
void wordpiece_tokenizer(const char* word) {
    int start = 0;
    int len = 0;
    while (word[len] != '\0') len++;

    while (start < len) {
        bool matched = false;
        int end = len;

        while (end > start) {
            char token[50];
            bool is_prefix = (start > 0);
            substring(word, start, end, token, is_prefix);

            if (in_vocab(token)) {
                cout << token << " ";
                start = end;
                matched = true;
                break;
            }
            end--;
        }

        if (!matched) {
            cout << "[UNK] ";
            break;
        }
    }
    cout << endl;
}

int main() {
    const char* test = "extraordinaire";
    cout << "Tokenisation de : " << test << endl;
    wordpiece_tokenizer(test);
    return 0;
}
```

Listing 1.1: Tokeniseur WordPiece en C++

Chapitre 2

Contexte et Problématique

Chapitre 3

Méthodologie

Chapitre 4

Résultats

Chapitre 5

Conclusion et Perspectives

Annexe A

Annexe A : Informations complémentaires