République Algérienne Démocratique et Populaire Ministère
de l'Enseignement Supérieur et de la Recherche Scientifique
Université A. Mira de Béjaïa
Faculté des Sciences Exactes
Département d'informatique

Mémoire de fin d'étude

En vue de l'obtention du diplôme de Master en Informatique

Option : Génie Logiciel (GL)

Thème

Conception et réalisation d'un traducteur Français –Tamazight sous Android.

Encadrés par :

Mme. Aloui Soraya.
Mme. Yaici Malika.

Réalisé par :

Mr Amrane Hocine.
Mr Amri Salim.

Devant le jury composé de :

Examinateur : Mr. Akilal Karim
Examinateur : Mr. Bedjou Khaled

2020/2021

Dédicaces

C'est avec profonde gratitude et sincères mots,

Que nous dédions ce modeste travail de fin d'étude a

Nos chers parents, qui ont sacrifié leur vie pour

Notre réussite et nous ont éclairé le chemin par

Leurs conseils judicieux.

Nous espérons qu'un jour,

Nous pourrons leur rendre un peu de ce qu'ils ont

Fait pour nous, que dieu leur prête bonheur et longue vie.

Nous dédions aussi ce travail à nos frères et

Sœurs, nos familles, nos amis,

Tous nos professeurs qui nous ont enseigner

Et à tous ceux qui nous sont chers

Remerciements

Nous remercions Dieu le tout puissant et miséricordieux qui nous a donner la force et la patience d'accomplir ce Modest travail.

Nous tenons à remercier toute les personnes qui ont contribué, d'une manière ou d'une autre à ce que modeste travail puisse aboutir

Nous tenons tout d'abord à adresser nos plus profonds et sincères remerciements à nos encadreur Mme Aloui Soraya et Yaici Malika.

Qui ont dirigé ce travail, pour tous leur conseils et encouragements,
pour leur disponibilité et compréhension.

Nous exprimons notre gratitude aux membre de jury d'avoir consacré le temps qu'il fallait pour lire et corriger ce mémoire et de l'honneur qu'ils nous ont fait pour en participer à notre soutenance

Enfin, on remercie tous ceux ou celles qui ont agi

dans l'ombre et participé discrètement à la réalisation

de ce mémoire.

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

DAML : Digitally Added Main Line.

XML : Extensible Markup Language.

XOL : Ontology Exchange Language.

OML : Ontology Markup Language.

HTML : Hypertext Markup Language.

RDF : Resource Description Framework.

W3C : World Wide Web Consortium.

OWL : Web Ontology Language.

UML : Unified Modeling Language.

EDI : Environnement de Développement Intégré.

Introduction générale

Introduction générale

La traduction est une pratique qui favorise le contact entre les peuples et contribue à l'échange culturel et littéraire entre ces derniers. Les amazighs ont connu cette pratique depuis l'antiquité, les stèles libyques bilingues et les manuscrits qui remontent au Moyen-Âge témoignent sur l'ancienneté de cette pratique dans le monde amazigh, depuis l'apparition de la revendication culturelle amazighe dans les années soixante, avec la naissance d'Agraw Imazighen (Académie Berbère) en France, des militants ont été conscients de l'importance que peut jouer la traduction dans l'enrichissement et le développement de la littérature amazighe. C'est à cette époque qu'apparut Mohia par ses adaptations kabyles du théâtre universel. Mohia ou Mohand Ouyahia, fut l'un des rares traducteurs et militants à avoir consacré toute sa vie à traduire les meilleures œuvres littéraires en particulier les pièces de théâtre. [1]

La traduction automatique est en plein essor sur Internet. Il existe plusieurs systèmes multilingues. Mais nous avons remarqué que pour notre langue, qui est tamazight, les moyens de traduction vers d'autres langues sont peu fréquents. C'est pourquoi nous avons pensé à créer un traducteur français-tamazight.

Avec l'aide d'un autre travail qui traite la direction tamazight vers le français, nous avons décidé de faire la traduction du français vers tamazight. [2]

L'objectif principal de ce travail est de contribuer à la construction d'un traducteur français-tamazight basé sur une ontologie. Ce travail est structuré comme suit :

Dans le premier chapitre, nous présentons les concepts de la langue tamazight et ses principales caractéristiques et les concepts de la langue française. Où nous verrons toutes les notions nécessaires et basiques pour connaitre ces deux langues.

Le deuxième chapitre est consacré aux ontologies, les stratégies de construction, les langages utilisés et la présentation de quelques éditeurs d'ontologies. Ensuite les traducteurs automatiques modernes existant.

Dans le troisième chapitre, nous présenterons l'implémentation de l'ontologie (français).

Le quatrième chapitre servira à la conception et à la réalisation de notre application où nous présenterons quelques diagrammes UML, ainsi que les outils utilisés. Nous détaillerons, par la suite, le fonctionnement de notre application à travers quelques captures d'écran et algorithmes.

Nous terminerons par une conclusion et des perspectives.

Chapitre 1 : Linguistique

Chapitre 1 : Linguistique

1. Introduction

L'attitude des Imazighen envers la langue qu'ils parlent se situe d'abord sur le plan historique. En effet, à part quelques moments de l'histoire, les Imazighen n'ont pas assumé à part entière leur propre langue. Ils ont assisté à l'élimination progressive de leur langue par les différentes cultures dans plusieurs régions.« *Le citoyen amazighophone qui revendique le droit au respect de son identité ne connaîtra pas de promotion professionnelle dans l'administration ou les entreprises publiques* »[3]. Les communautés amazighophones ont un mode de vie déterminé par leur appartenance à l'ensemble culturel arabe ou français et la volonté de résister à la dominance du milieu arabophone ou francophone sur leurs parlers. La culture n'est pas séparée de la langue qui est menacée sur le terrain. Depuis 1980, la société amazighe, surtout kabyle est entrée dans une ère de transformations profondes et accélérées pour la définition de soi, comme individu ou groupe.[4]

2. Système d'écriture tamazight

Bien que tamazight était principalement une langue de tradition orale, les amazighophones possèdent, depuis l'Antiquité, leur propre système d'écriture appelé «Libyco-berbère » (le Tifinaghe en amazighe). Ce système est de nature alphabétique, à tendance phonologique fondée sur des signes à valeur consonantique, il date de plus de 40 siècles, on utilise trois alphabets (arabe, latin, tifinagh) pour transcrire le berbère, en Algérie. On emploie au même temps les trois, avec une nette prédominance du «latin». Dans le cadre de notre projet nous nous focaliserons sur l'alphabet latin .[4]

3. Introduction de tamazight

Le berbère ou tamazight, réalisée de différentes façons selon les régions, tamazight ne s'exprime et ne se pratique qu'à travers les dialectes qui la composent (chaoui, chleuh, kabyle, rifain, touareg, etc.). l'alphabet de la langue tamazight est composée de 33 lettres comme suit :

[**a,** أ, ⵄ], [**b**, ب, ⵀ], [**c,** ش, ⵛ], [**č,** تش, ⵜⵛ], [**d,** د / ذ, ⴷ], [**ḍ,** ض / ظ, ⴹ], [**e,** ⵄ, ء], [**ɛ,** ع,ⵃ],[**f,** ف,ⴼ], [**g,** (گ), ⴳ],[**ǧ,** ج, ⴶ], [**ɣ/gh,** غ, ⵖ], [**h,** ه , ⵀ], [**ḥ,** ح, ⵃ], [**i,** ي, ⵉ], [**j,** ج, ⵊ], [**k,** ک,ⴽ],[**l,** ل,ⵍ], [**m,** م, ⵎ], [**n,** ن, ⵏ], [**q,** ق, ⵇ], [**r/ř,** ر,ⵔ], [**s,** س, ⵙ],[**ṣ,** ص, ⵚ], [**t,** ت / ث, ⵜ], [**ṭ,** ط, ⵟ],[**u,** و / ُ, ⵓ], [**w,** ۇ, ⵡ], [**x,** خ, ⵅ], [**y,** ۑ, ⵢ], [**z,** ز ,ⵣ], [**ẓ,** ژ, ⵥ].[5]

4. Morphologie nominale de la langue tamazight

La langue tamazight présente une morphologie riche et complexe. Les mots peuvent être classés en quatre catégories morphosyntaxiques : Nom, adjectif, adverbe et le verbe . [6]

4.1 Nom

Dans tamazight, le nom est une unité lexicale formée d'une base et d'un ou plusieurs affixes. Cette base résulte de la combinaison d'une racine et d'un schème, Le nom possède deux caractéristiques.

a) La première caractéristique : le nom peut prendre différentes formes à savoir : une forme simple ([argaz] homme), une forme composé ([aɣesmar] iɣes « os » + amar « menton ») ou bien une forme verbale dérivée ([kcem] entrer -> [ssekcem] introduire).

- **Noms simples**

Tamazight distingue deux grandes sous –classes de noms simples : les noms propres et les noms communs

- **Noms propres**

La règle	Exemple
Les noms propres désignent soit des personnes ou des lieux.	Hamid, Karim, Bâb lbher, Akbou.

Tableau 01 : Les noms propres Tamazight.

- **Noms communs**

La règle	Exemple
Les noms communs peuvent être soit abstraits soit concrets. Ces derniers peuvent, à leur tour, être animés ou inanimés.	animé : tamṭut(femme), izem(lion). inanimé axxam(maison).

Tableau 02 : Les noms communs Tamazight.

- **Noms dérivés :**

A partir d'une racine verbale, les noms dérivés sont formés par une préfixation ou suffixation d'un morphème (forme minimum douée de sens (mot simple ou élément de mot)) de dérivation plus des variations intra-radicales. Le nombre et la nature de ces formes varient selon le statut du verbe auquel ils se rattachent. Ainsi, le nom d'action, le nom d'agent, le nom d'instrument et le nom de qualité sont constitués. [6]

- **Noms d'actions**

La règle	Exemple
Le nom d'action se forme avec des préfixes associés à des modifications intra-radicales. Les principaux procédés de dérivation sont : préfixation de [a], [u], [i], préfixation et affixation du morphème du féminin [t….t].	nom d'action = préfixation + verbe ([zdem] attaquer ->[azdem] s'attaquer)

Tableau 03 : Les noms d'actions Tamazight.

- **Nom d'agent**

La règle	Exemple
Le nom d'agent (celui qui fait l'action) dérive d'un verbe d'action par préfixation de l'un des éléments suivants [a], [am], [an][im], [i].	Nom d'agent=préfixation+verbe :([akr] voler->([amakar] voleur), ([ttef] tenir ->[anattef] celui qui tient)

Tableau 04 : Les noms d'agent Tamazight.

- **Nom d'instrument**

La règle	Exemple
Ce type de nom est formé sur la base des schèmes [a], [am],[as] associé à des modifications vocaliques ou consonantiques.	([nqer] percer->[amenqar]burin).

Tableau 05 : Les noms d'instrument Tamazight.

- **Nom de qualité**

La règle	Exemple
Le nom de qualité est généralement dérivé des verbes dits de qualité ou d'état. Les procédés de formation de ce type de nom sont préfixation de [a],[am],[an] suivi d'une variation intra ou post-radicale.	.

Tableau 06 : Les noms de qualité Tamazight.

b) La deuxième caractéristique : Un nom Tamazight correspond à la flexion : il varie en genre (féminin, masculin), en nombre (singulier, pluriel) et en état (libre, annexion). [7]

- **Genre**

- **Masculin**

La règle	Exemple
Le nom masculin commence généralement par une des voyelles initiales :[a],[i] ou bien [u].	([udem] visage, [aserdun] cheval, [asennan] épine). il existe certains noms qui font exception comme : ([baba] mon père).

Tableau 07 : Masculin Tamazight.

- **Féminin**

La Règle	Exemple
Celui-ci est généralement de la forme [t….t], a l'exception de certains noms qui ne portent que [t] initial ou le [t] final du morphème du féminin.	([isli]marié->[tislit]mariée, [alus] beau-frère->[talust]belle-Sœur).

Tableau 08 : Féminin Tamazight.

- **Nombre :** le nom en Tamazight, qu'il soit masculin ou féminin, possède un singulier et un pluriel. Ce dernier est obtenu selon quatre types : le pluriel externe, interne, mixte et le pluriel en [at].

- **Pluriel externe**

La Règle	Exemple
Obtenu par une alternance vocalique accompagnée par une suffixation pour le masculin [en], [an], [wen],et pour le féminin([in].	axxam->[ixxamen]maisons, [udem]->[udmawen]visages). [taxxamet->tixxamin).

Tableau 09 : Pluriel externe Tamazight.

- **Pluriel interne**

La Règle	Exemple
Obtenu par une alternance vocalique plus un changement de voyelles internes.	([adrar]->[idurar]montagnes).

Tableau 10 : Pluriel interne Tamazight.

- **Pluriel mixte**

La Règle	Exemple
Formé par une alternance d'une voyelle interne et/ou d'une consonne plus une suffixation par [n],ou bien par une alternance vocalique initiale accompagnée d'un changement vocalique final [a] plus une alternance interne.	([anggaru]->[inggura]derniers)

Tableau 11 : Pluriel mixte Tamazight.

- **Pluriel en [at]**

La Règle	Exemple
Obtenu avec une suffixation [at] du nom au singulier. Il est appliqué à l'ensemble des noms empruntés et intégrés.	([lbabur] bateau->[lbaburat]bateaux).

Tableau 12 : Pluriel en [at] Tamazight.

➢ **L'état**

En Tamazight, le nom est concerné par la variation d'état. Ainsi, nous distinguons deux états: libre et d'annexion.

- **L'état libre**

La Règle	Exemple
La voyelle initiale du nom ne subit aucune modification. Le nom est en état libre lorsqu'il s'agit : d'un mot isolé de tout contexte syntaxique, où d'un complément d'objet direct.	([amɣar] vieux, [aqjun] chien).

Tableau 13 : Etat libre Tamazight.

- **L'état d'annexion**

La Règle	Exemple
Est fondé sur une variation formelle qui affecte la première syllabe des noms en cause dans des contextes syntaxiques déterminés. Alternance vocalique [a] /[u], ou bien maintien de la voyelle initiale et ajout d'un [w] au cas des noms masculins à initiale [a], addition d'un [w] pour ceux a initial [u], pour les noms féminins, cet état est défini soit par la chute ou le maintien de la voyelle initiale.	([argaz]->[iruh wargaz ɣer ljam3]), ([ils] langue [yils]), ([izem]lion -> [allen n yizem])

Tableau 14 : Etat libre Tamazight.

4.2 Adjectif

L'adjectif est formé avec préfixation ([a], [u], [i], [[an], + nom, [ans] + nom, [am] + nom) : l'adjectif est connu dans la plus grande partie de l'ensemble linguistique berbère, il est formé sur un radical verbal à quelques exceptions prés.[7]

L'adjectif partage tous les traits combinatoires et fonctionnels du substantif. Il porte, lui aussi, les marques de :

Genre	Nombre	Etat
([amellal]blanc,[tamellalt] blanche,).	([amellal]blanc, [imellalen]blancs],).	([amellal]->[umellal).

Tableau 15: Adjectif Tamazight.

4.3 Adverbe

Les adverbes sont des déterminants (expansions facultatives) ayant comme particularité de n'être reliés au reste de l'énoncé par aucun indicateur de fonction explicite. [7]

Adverbes de manière	adverbes de temps	adverbes de Lieu	adverbes de Quantité
[kifkif=même, également], [yak=n'est-ce pas].	[zik=tôt,[ass-n=cejour la],[Ilindi=l'année passée],[tanezzayt=ce matin].	:[da,dagi,dayi=ici], [tama=a côté de].	[kra, cra,, drus=un peu], [bezaf=trop], [anect=autant].

Tableau 16 : Adverbe Tamazight.

4.4 Verbe

Dans une forme verbale amazigh, on trouve obligatoirement : une racine lexicale, une marque aspectuelle ou schème et un indice de personne.

On peut ajouter facultativement des marques dérivationnelles concernant la voix ou la diathèse le factitif, le passif et le réciproque.

Exemple : la forme verbale du verbe[yezra], se décompose comme suit :

y : indice de troisième personne de masculin.

zr : racine lexical.

a :marque de l'aspect accomplit.

- **Dérivation Verbale**
  - **Sur base verbale**

- Factitif avec préfixation [s] : factitif est une forme verbale qui indique que le sujet fait faire l'action par un autre agent que lui-même.
  Exemple : ([rwel] fuir, [serwel] faire fuir) .
- Passif avec préfixation [ttw], [m], [n] : passif est une forme verbale qui indique que le sujet subit l'action exprimée par le verbe au lieu de la faire.
  Exemple : ([kkes] ôter, [ttwekkes /mekes] être ôté).
- Réciproque avec préfixation [my], [mi], [mm] : l'action de X envers Y est la même que celle de Y envers X. Une construction de cet ordre est dite réciproque.
  Exemple : ([myukkas] se dérober).

  - **Sur base nominal**
    - Préfixation de [s-] : exemple :[smuhbel](faire le fou), [suden](embrasser sur le visage),
  - **Composition :** C'est la formation d'une nouvelle unité lexicale à partir de deux unités lexicales autonomes
    - Verbe + nom, nom + verbe :[amagraman](mager = rencontrer + aman = eau)
    - Verbe + verbe :[bbirwel](bbi = pincer + rwel =se sauver)
    - Préposition + nom :[agersif](ger = entre + asif = fleuve)
    - Nom d'action verbal + son naturel :[amqarqur](amqar = celui qui dit + qur = son de grenouille)[8]

4.5 Interrogation

- **L'interrogation directe**

Interrogation directe comporte fréquemment un mot interrogatif et se termine toujours à l'écrit par un point d'interrogation [9]

Mot interrogatif : [ma,mec] est-ce-que

Exemple :[ma dderren ? es-ce-que il sont vivants ?] , [ma tejjit ? es-tu guéri ?].

Mot interrogatif :[milmi] quand

Exemple [milmi iγra adlis ? quand a-t-il lu le livre]

- **L'interrogation indirecte**

Les questions indirectes sont des phrases, où la structure interrogative est incluse dans une autre proposition principale.

Exemple : [ina-s ma tusa-d yemma-s ?] demande-lui si sa mére est arrivée ?

L'intonation interrogative peut à elle seule marquer l'interrogation :

Exemple :[yejji ?] il'est guéri ? , [d'amellal] il est blanc ?

4.6 La négation

Le morphème de négation en tamazight est [ur]. Il se place toujours devant un verbe, un nom, un prénom ou une préposition. En français, il correspond au morphème de négation discontinu « ne pas ».[9]

Exemple : ([yezzenz-> ur yezzenz], il n'a pas vendu),([yeffud->ur yeffud il n'a pas soif].

5. Introduction à la langue française

Le français est une langue indo-européenne de la famille des langues romanes dont les locuteurs sont appelés francophones. Ces locuteurs sont repartis sur tous les continents on estime que 300 millions de personnes, et 90 millions en sont des locuteurs natifs. [10]

5.1 Morphologie française

La langue française se constitue des mots, de nature qui varient (nom, verbe, adjectif) et d'autres qui ne varient pas (préposition, conjonction, adverbe).

5.1 Nom

Les types des noms dans la langue française sont plusieurs et classés comme suit :

- **Nom propre**

La règle	Exemple
Le nom propre s'écrit avec une lettre majuscule. Il s'applique à une personne, à un lieu, une chose unique.	(Hocine, Romaissa, Alger).

Tableau 17 : Nom propre Français.

- **Nom commun**

La règle	Exemple
Le nom commun s'écrit avec une lettre minuscule, il correspond à un concept et s'oppose au nom propre qui lui n'a pas de signification véritable.	(une chambre [taxxamt], un bureau[asirra]).

Tableau 18 : Nom commun Français.

- **Nom abstrait**

La règle	Exemple
Le nom abstrait désigne une propriété, une qualité, une relation. Il ne désigne pas les objets du monde.	(l'imagination [amagur], la pensée[axemem]).

Tableau 19 : Nom abstrait Français.

- **Nom composé**

La règle	Exemple
Le nom composé est un mot formé d'unités lexicales qui fonctionnent de manière autonome dans la langue.	(porte-monnaie[tatezdamt],après-midi [tamedit]).

Tableau 20 : Nom composé Français.

Le nom varie en genre (masculin, féminin) et de nombre (singulier, pluriel) [11]

Féminin

Pour transformer un nom au féminin, en général on rajoute « e » à la fin de mot. D'autre exception quand il se termine par (er) on le remplace par «ère »,le cas (en,on)-> «nne », (eur)-> « euse,trice ».

Exemple : (un voisin une voisine, un lion une lionne, un coordinateur une coordinatrice).

Pluriel

Pour transformer un nom au pluriel en général on rajoute « s » à la fin de mot. D'autre exception comme quand il se termine par (al) on le remplace par «aux »,le cas (au,eau,eu)-> «x », (ou)-> « s ».

Exemple (un voisin des voisins, un cheval des chevaux).[12]

5.2 Adjectif

L'adjectif ne peut pas se constituer avec le nom commun ou un groupe nominal sans un déterminant (le, la. .). L'adjectif présente une propriété d'un substantif, s'accorde en genre et en nombre avec le nom.

Exemple **:**(un chat blanc avec des poils longs et des yeux bleus)

5.3 Verbe

Un verbe est un mot qui exprime une action, un état, un fait ou une intention, Il peut être régulier ou irrégulier. Il se conjugue et varie, comme suit

- **Mode :**

  Le mode personnel conjugué (indicatif, impératif [un mode utilisé pour donner un ordre ou un conseil à une ou plusieurs personnes], subjonctif [un mode utilisé pour exprimer un doute, un fait souhaité, une action incertaine qui n'a donc pas été réalisée au moment où nous nous exprimons.]).

  Le mode impersonnel (infinitif [est une forme non conjuguée du verbe], le participe [est un mode du verbe qui lui donne les caractéristiques d'un adjectif]).

- **Temps :** présent future le passé simple. Exemple : (il mange, il mangera, il mangea)
- **Voix :** voix active exemple (Sara rédige un texte), voix passive exemple :(Le texte est rédigé par Sara).
- **Nombre :** singulier, pluriel. Exemple :(Elle riait Elles riaient).
- **Genre :** masculin, féminin. Exemple :(blessé - blessée).[12]

5.4 Adverbe

L’adverbe est un mot invariable qui précise ou change le sens d'un verbe, d'un adjectif, ou d'un autre adverbe. Les différentes catégories d'adverbes :[12]

- **Adverbes de manière** : ainsi, bien, comme, comment, debout.
- **Adverbes de quantité** : assez autant, beaucoup, combien.
- **Adverbes de temps :** alors, après, après-demain, aujourd'hui.
- **Adverbes de lieu :** ailleurs, alentour, arrière, autour.
- **Adverbes d'affirmation :** assurément, aussi, certainement, bien.
- **Adverbes de négation :** non, aucun, aucunement, nullement.
- **Adverbes de doute :** apparemment, peut-être, probablement, sans.

6. Conclusion

En conclusion, le choix et l’enseignement du lexique de tamazight dépend, de la place des langues minoritaires et des traditions orales sur l’échiquier linguistique. En particulier Tamazight, mais il dépend, également, de la volonté de ses utilisateurs à élever cette langue au rang des langues d’études. Dans ce travail nous avons étudié les ressources linguistiques pour Tamazight et Français. Pour ensuite l’implémenter dans notre réalisation d’une application d’un traducteur Français Tamazight.

Chapitre 2 : Introduction aux ontologies

Chapitre 2 : Introduction aux ontologies

2.1 Les ontologie

L'ontologie est un terme d'origine philosophique. Dans la philosophie, l'ontologie s'intéresse à l'étude de l'être, c'est-à-dire, une étude des propriétés générales de ce qui existe. Cependant, en informatique et d'après l'encyclopédie Wikipédia, une ontologie est un ensemble structuré de concepts. Les concepts sont organisés dans un graphe. Dont les relations peuvent être sémantiques où des relations de composition et d'héritage (au sens objet). [13].

Le terme d'ontologie a été créer dans les débuts des années 1990 dans les domaines de l'intelligence artificielle, en particulier celui de l'ingénierie et de la représentation des connaissances. Son champ d'application s'accroit considérablement et il fait désormais partie des objets de recherche courants. Une ontologie est un système formel, son objectif est de représenter les connaissances d'un domaine spécifique avec des concepts, définis et organisés les uns par rapport aux autres [14].

Dans ce chapitre, nous passerons en revue les différentes définitions qui ont été attribuées à la notion d'ontologie, nous verrons aussi les éléments dont elle est composée, ainsi que les outils qui permet à sa manipulation.

2.2 Notion d'ontologie

Les ontologies définissent des vocabulaires structurés, regroupant des concepts utiles d'un domaine et de leurs relations et qui servent à organiser et échanger des informations de façon non ambiguë. Afin de facilité le partage des connaissances entre les individus. [15].

2.3 Définitions

Le terme ontologie a connue beaucoup de définitions en cours des années. Concéderons la difficulté de le définir d'une façon définitive. Nous allons présenter les définitions suivantes :

- **Définition 1** : *une ontologie définit les termes et les relations de base du vocabulaire d'un domaine ainsi que les règles qui indiquent comment combiner les termes et les relations de façon à pouvoir étendre le vocabulaire*. –Neches-

Neches proposa cette définition afin de montrer comment l'élaboration des ontologies s'effectue, mais la définition reste toujours floue : repérer les termes de base et les relations entre les termes, identifier les règles servant à les combiner, fournir des définitions de ces termes et de ces relations. [16]

Notons que d'après cette définition, une ontologie inclut non seulement les termes qui y sont explicitement définis, mais aussi les termes qui peuvent être crées par déduction en utilisant les règles.

- **Définition 2** : *une ontologie est une spécification explicite et formelle d'une conceptualisation partagée*. –Gruber-

Spécification formelle : compréhensible par une machine. Spécification explicite : les concepts, relations, fonctions, contraintes, axiomes sont explicitement définis, Partagés : les connaissances

représentées sont partagées par une communauté. Conceptualisation : modèle abstrait d'une partie du monde que l'on veut représente

Pour nous, la deuxième définition est plus spécifique et répond à nos besoins. Les objets(mots), les relations entre les mots sont définies explicitement. Ainsi nous conceptualisons notre ontologie en se basant sur cette définition.

2.4 Le rôle des ontologies

Les ontologies sont utilisées pour résoudre deux grands problèmes qui sont l'interopérabilité et la réutilisation [17] :

- **L'interopérabilité :** Les ontologies présentent une solution face aux ambiguïtés des langages utilisés, car elle assure la cohérence en éliminant la confusion terminologique concernant les termes utilisés.
- **La réutilisation :** l'ontologie utilise le moyen de réutilisation des concepts et des méthodes déjà établies.

2.5 Les éléments constituant une ontologie :

Une ontologie est constituée principalement de Concepts, des Relations, des Axiomes, des Instances [17] :

• Concepts :

Ils sont les éléments fondamentaux d'une ontologie. Ils représentent des classes ou un groupe d'objets, dans un domaine, qui partage des propriétés communes. Par exemple classe « personne »

• Relations :

Les relations d'une ontologie désignent les différentes interactions existant entre les concepts présents dans le domaine. Par exemple la classe « père » est une sous-classe de « personne ».

• Axiomes :

Les axiomes servent à définir le sens des entités, mettre des restrictions sur la valeur des attributs, examiner la conformité des informations spécifiées ou en déduire de nouvelles. Par exemple pour qu'une « personne » soit un « père » il faut qu'il a des « enfants ».

• Instances :

L'instance est un objet occurrence de concept. Par exemple les individus « Amina » et « Saloua » sont des instances du concept « personne ».

2.6 Différentes types d'ontologies :

Lors de la construction ou du choix d'une ontologie, il est important d'avoir à l'esprit que plus le niveau d'abstraction choisi est proche de l'application moins l'ontologie est réutilisable, mais plus elle est utilisable, différents types d'ontologies sont développés :

• Ontologie de domaine :

Les ontologies de domaine se caractérisent d'être plus spécifiques. Elles représentent un vocabulaire qui est relié à un domaine unique. En spécialisant les concepts introduits dans l'ontologie de haut niveau [16].

• Ontologie de tâches :

Ce type d'ontologie est utilisé pour conceptualiser des tâches spécifiques dans des systèmes qui peuvent être ou non du même domaine, telles que les tâches de diagnostic, de planification, de conception, de configuration. Soit tout ce qui concerne la résolution de problèmes. [16].

• Ontologie d'application :

Cette ontologie est la plus spécifique, elle contient des concepts dépendants d'un domaine particulier et d'une tâche particuliere. Ces concepts non réutilisables correspondent souvent aux rôles joués par les entités du domaine lors de l'exécution d'une certaine tache (ex. : apprendre les statistiques, effectuer des recherches dans le domaine de physique, etc.) [16].

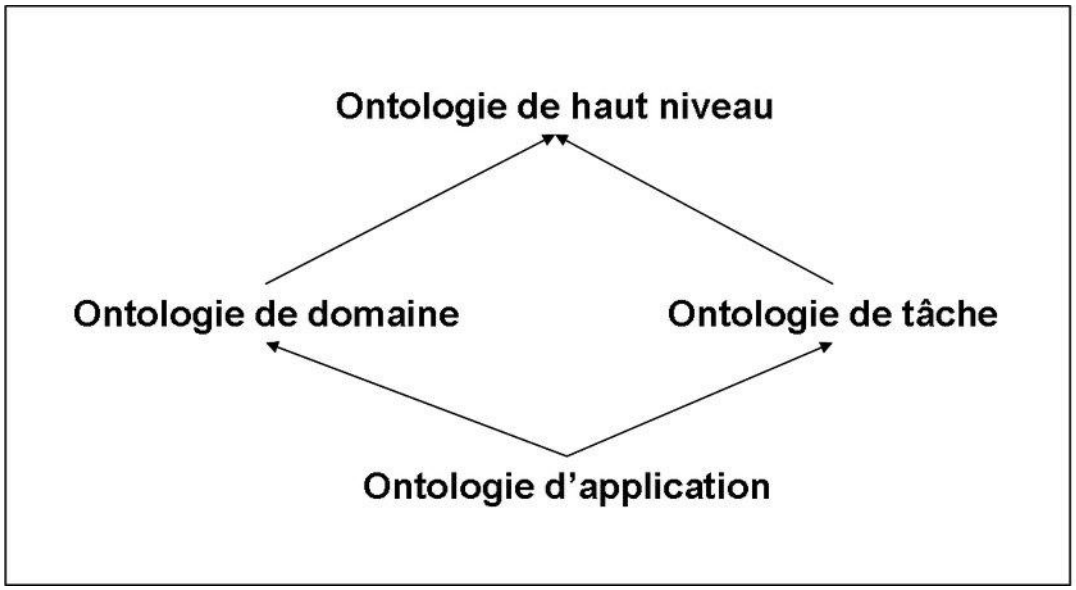

Figure 01 : Types d'ontologie.

2.7 Quelques méthodologies de construction d'ontologies :

L'objectif des méthodologies est d'aider et faciliter la construction de l'ontologie. ENTERPRISE et METHONTOLOGY sont les méthodologies les plus connue.

2.7.1 Enterprise :

Uschold et King's [19] proposent le squelette d'une méthode basée sur l'expérience de construction d'ontologies dans le domaine de la gestion des entreprises. La méthode ENTERPRISE repose sur les étapes suivantes :

- La description de l'entreprise.
- Spécifications des exigences relatives aux problèmes d'entreprise.
- Identification et évaluation des options de solution et des conceptions alternative sur le niveau opérationnel.
- Faire un outil de support et une documentation sur le travail.

2.7.2 Methontology :

Cette méthode est développée au laboratoire d'intelligence artificielle de l'Université polytechnique de Madrid. Elle vise la construction d'ontologie au niveau de connaissance. L'approche METHONTOLOGY repose sur les étapes suivantes. [20]

- L'identification du processus de développement de l'ontologie.
- Le cycle de vie basé sur l'évolution de prototypes.
- Les techniques de gestion de projet (planification, assurance qualité), de développement (spécification, conceptualisation, formalisation, implémentation, maintenance) et des activités de support (intégration, évaluation, documentation).

2.8 Les Langages de spécification d'ontologie :

Plusieurs langages d'ontologies ont été développés pendant les dernières années. Certains d'entre eux sont basés sur la syntaxe de XML, tels que XOL (Ontology Exchange Language), OML (Ontology Markup Language) basés sur HTML, RDF (Resource Description Framework), RDF Schéma. Les deux derniers sont des langages créés par des groupes de travail du World Wide Web Consortium (W3C)

2.8.1 DAML-OIL :

DAML + OIL est un [1] langage de description basé sur XML pour les ontologies normalisées par le W3C. Après 2001, DAML + OIL n'a plus été développé. Le W3C a lancé le projet Web Ontology Language (OWL), qui est le successeur direct.[21]

2.8.2 OWL :

Web Ontology Language (OWL) est un langage de représentation des connaissances construit sur le modèle de données de RDF. Il fournit les moyens pour définir des ontologies web structurées. Sa deuxième version est devenue une recommandation du W3C fin 2012. OWL définit trois sous-langages. OWL6-Lite, OWL-DL, OWL-Full. [22]

2.9 Des éditeurs d'ontologies

Nous présentons quelques outils de construction d'ontologies.

- **Protégé-OWL**

PROTEGE-OWL est une interface modulaire, développée au Stanford Medical Informatics de l'Université de Stanford, permettant l'édition, la visualisation, le contrôle d'ontologies.

Notre choix portera sur cet outil pour faciliter l'implémentation de l'ontologie de notre application [23].

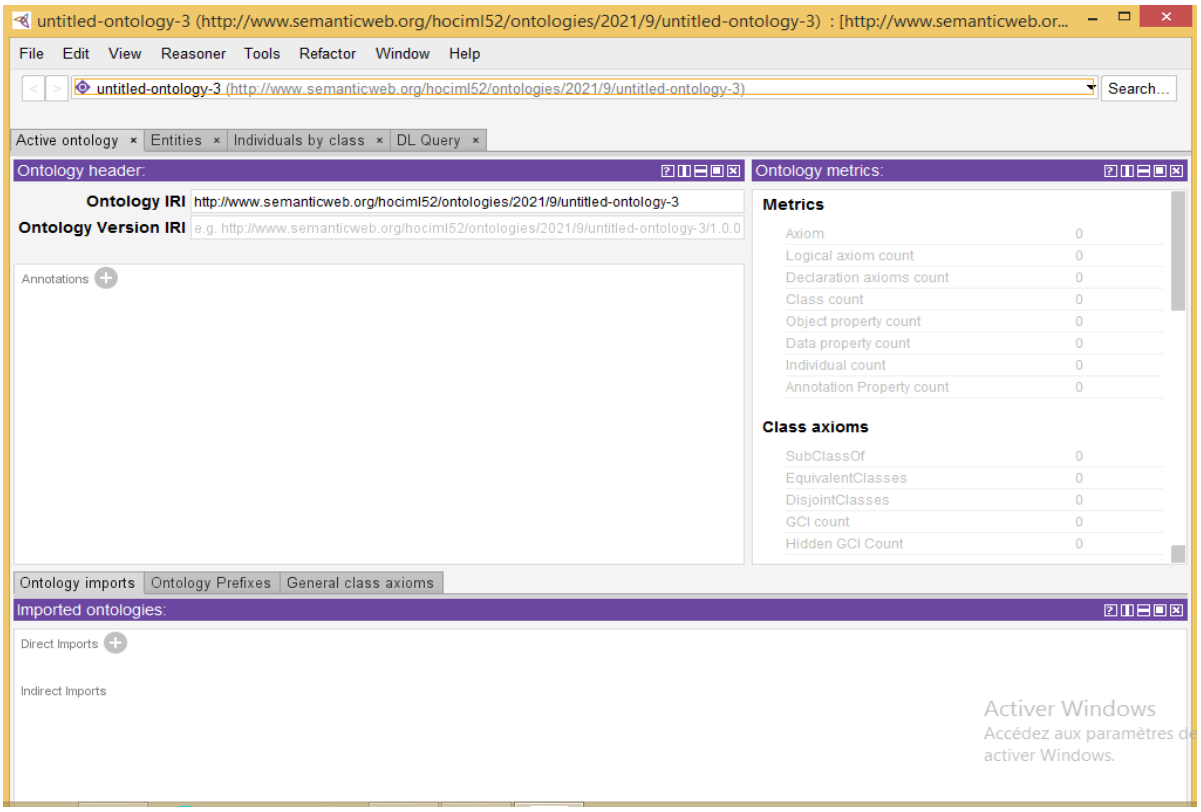

Figure 02 : Logiciel Protége.

- **ONTOEDIT (Ontology Editor)**

C'est un environnement équipé d'outils graphiques pour la visualisation et la construction d'ontologies indépendamment de tout formalisme.

2.10 Traducteur moderne

Nous avons remarqué que notre langue maternelle Tamazight fait face à un manque crucial dans le domaine de traduction automatique. On trouve aussi un manque de références et de données. Alors nous présentons les outils de traduction automatique les plus connues de nos jours.

2.10.1 Google Translate

Pour l'instant, Tamazight n'apparaît pas encore sur l'outil de traduction du moteur de recherche, mais elle est déjà proposée dans l'outil Google Translate Community, qui permet à n'importe quel utilisateur de contribuer à l'amélioration des traductions.

Figure 03 : Google translate.

2.10.2 Deepl

DeepL est un service de traduction automatique en ligne de la société DeepL GmbH, qui a été lancé le 28 août 2017 par l'équipe de Linguee. Le service permet de traduire vingt-six langues.[24]

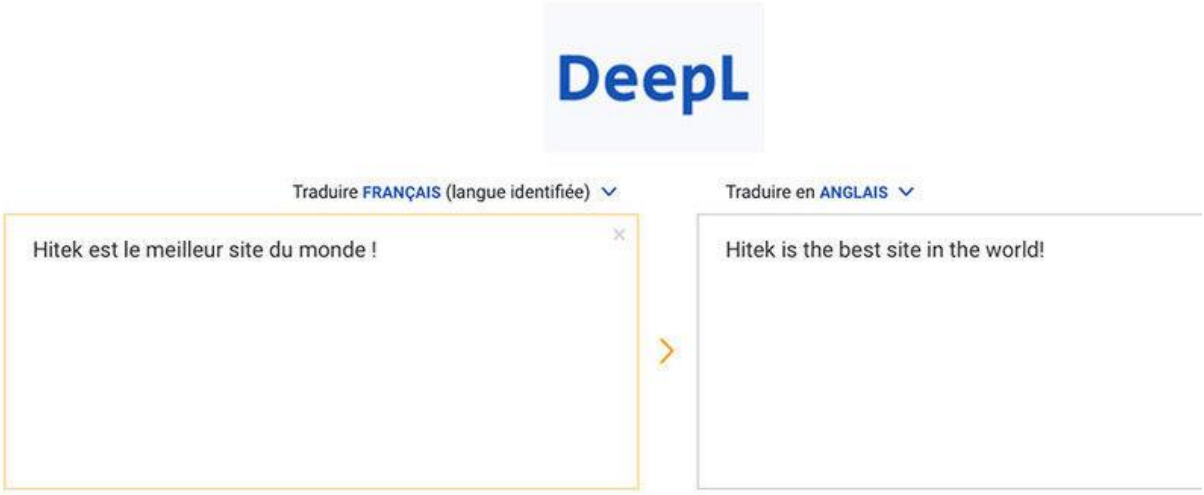

Figure 04 : Deepl.

2.11 Conclusion

Nous avons détaillé dans ce chapitre la notion d'ontologie, en présentant certaines définitions. Nous avons montré les méthodologies les plus représentatives de leur construction et quelques domaines de leur utilisation. Nous avons présenté les outils nécessaires à leur développement à savoir les langages de représentation. Finalement, quelques outils de traduction existants.

Chapitre 3 : Conception de L’ontologie

[1]Chapitre 3 : Conception de l'ontologie

3.1 Implémentation de l'ontologie

Nous présentons dans cette section l'ontologie utilisée, ainsi que quelques concepts offerts par le logiciel protégé voir chapitre 2.

3.1.1 Visualisation de l'ontologie

Protégé offre la fonctionnalité de visualiser l'ontologie sous forme de graphe avec OntoGraf comme suit.

- **Tamazight :**

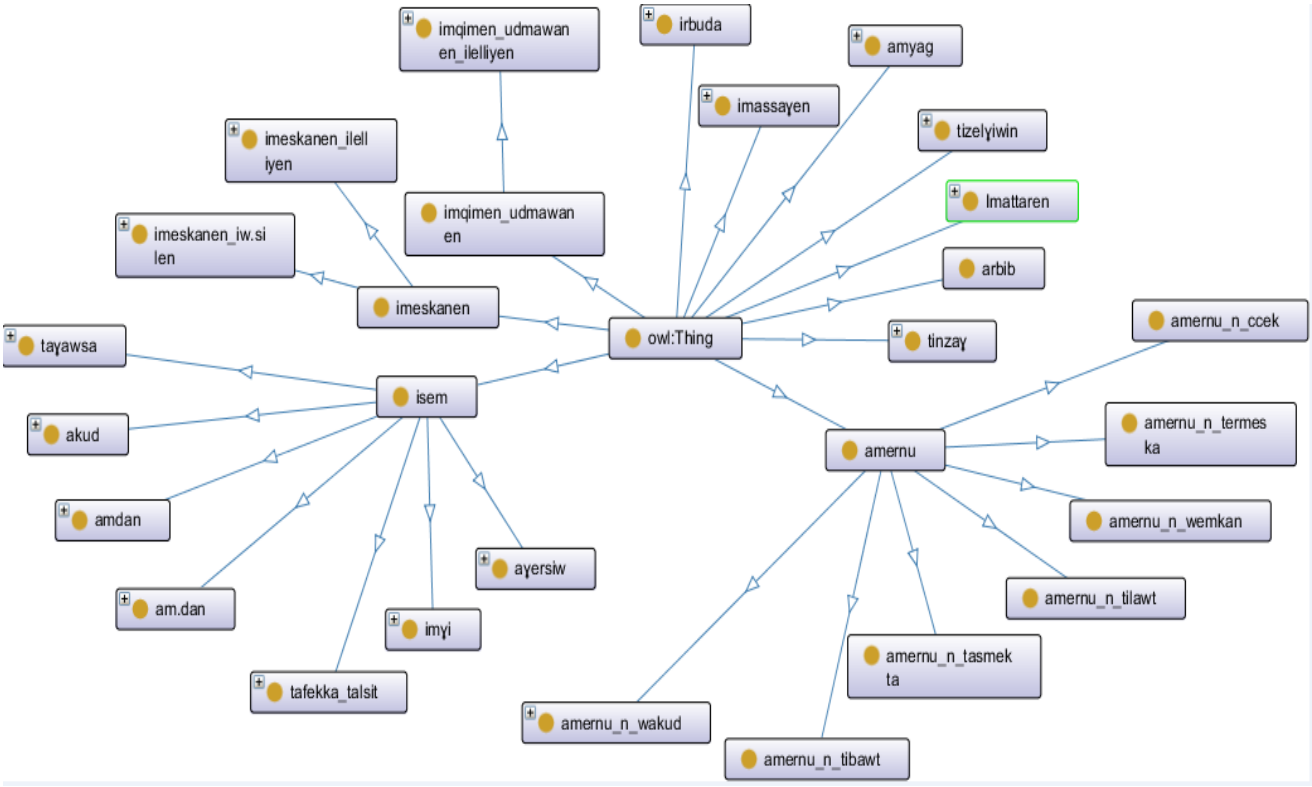

Figure 05 : visualisation du graphe d'ontologie de tamazight.

Dans la figure 05 on voit l'ensemble des classes de l'ontologie en langue tamazight, on cite quelques exemples (isem =nom),(irbuda=adjectif),(amyag=verbe), la figure 06 suivante montre ses mêmes classes mais dans la langue française.

[1] OntoGraph c'est une fonctionnalité de Protégé.

- **Français :**

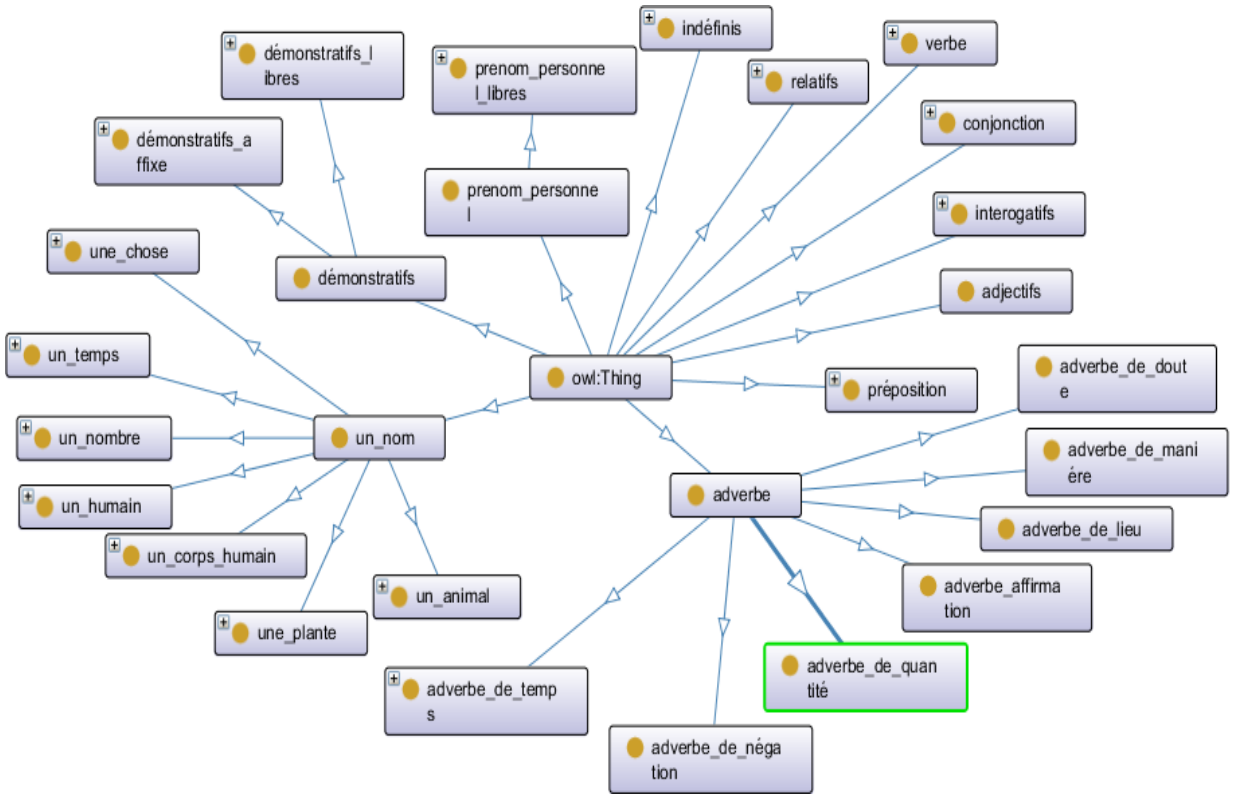

Figure 06 : visualisation du graphe d'ontologie de français.

3.1.2 Eléments de l'ontologie

Nous utiliserons l'ontologie précédente. Nous présenterons quelques statistiques de l'ontologie, obtenues par Protégé.

- **Français :**

Metrics

Axiom	5080
Logical axiom count	1807
Declaration axioms count	1576
Class count	1576
Object property count	0
Data property count	0
Individual count	0
Annotation Property count	2

Class axioms

SubClassOf	1572
EquivalentClasses	226
DisjointClasses	9
GCI count	0
Hidden GCI Count	393

Figure 07 : Statistique de l'ontologie française.

Dans la figure 07, nous présentons quelques statistiques de l'ontologie. Class count =>nombre de mot. Subclass of =>nombre mots avec la relations d'héritage. EquivalentClasses=>mots synonyme. Axiom=>nombre des relations.

3.1.3 Relations

Nous avons choisi de présenter quelques relations que Protégé offre afin de concevoir l'ontologie.

- **Subclass :** Cette relation désigne l'héritage entre les classes.

Dans la figure 08, le verbe « manger » a comme père la classe verbe pour signifier que manger est un verbe.

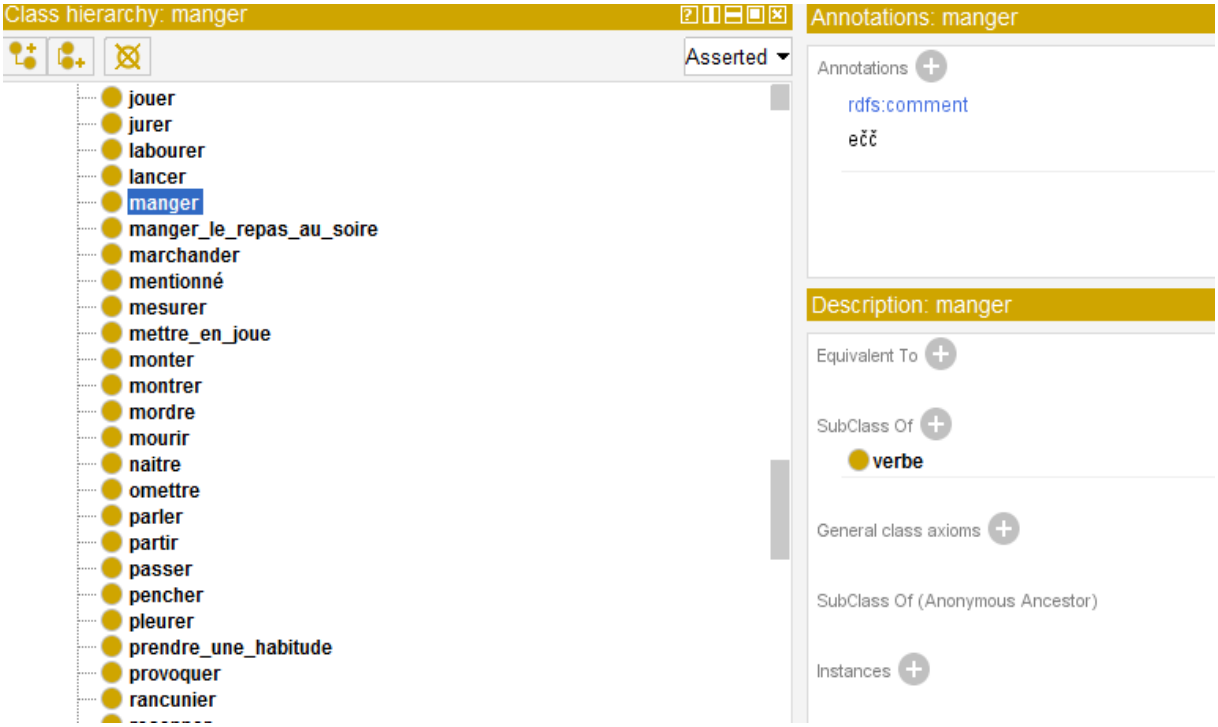

Figure 08 : Relation subclass.

- **Equivalent to :**

Elle désigne une relation d'égalité et de synonyme, représente deux mot ayant le même sens. Dans la figure 09, « lorsque » Equivalent To « quand ».

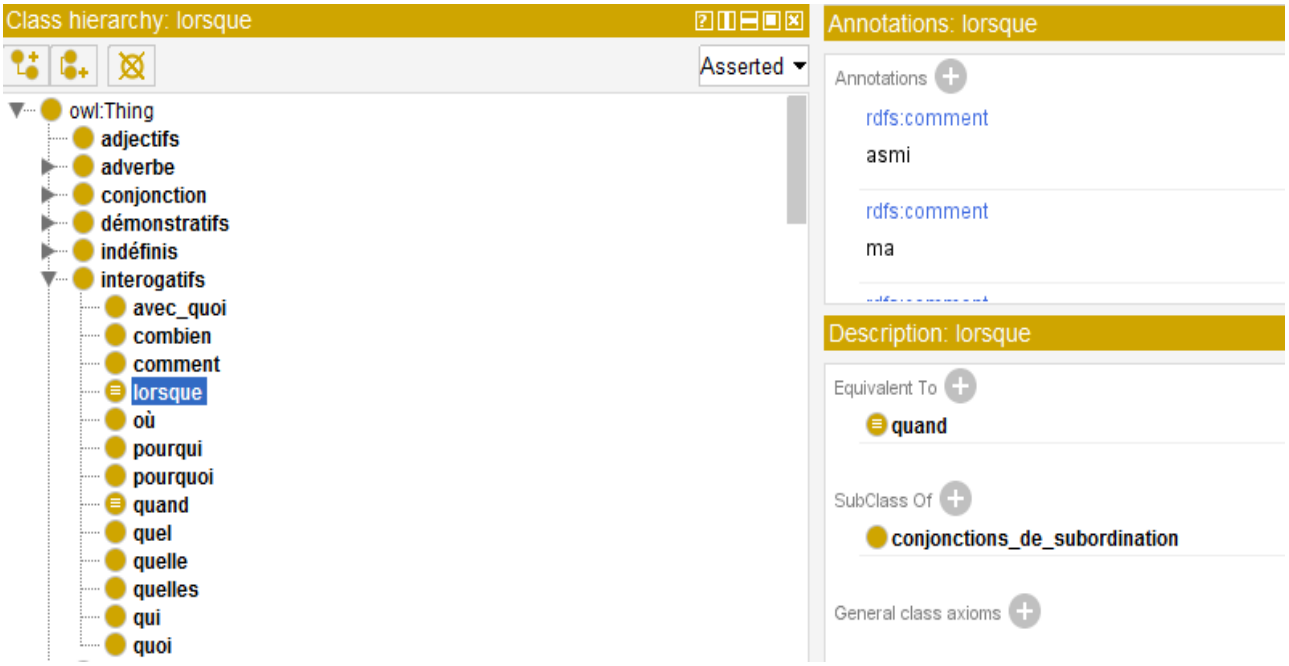

Figure 09 : Relation Equivalent to.

- **Les relations simples (Annotation) :**

Nous avons utilisé la partie Annotation de Protège pour associer à chaque mot français sa correspondance tamazight. La syntaxe d'écriture pour avoir cette association est écrite par le mot clé « rdfs:comment». Dans la figure 10, un homme rdfs:comment argaz. Donc le mot homme correspond en tamazight au mot argaz,

Dans la figure 11, chaise rdfs:comment : akersi, signifier que le mot chaise donne akersi.

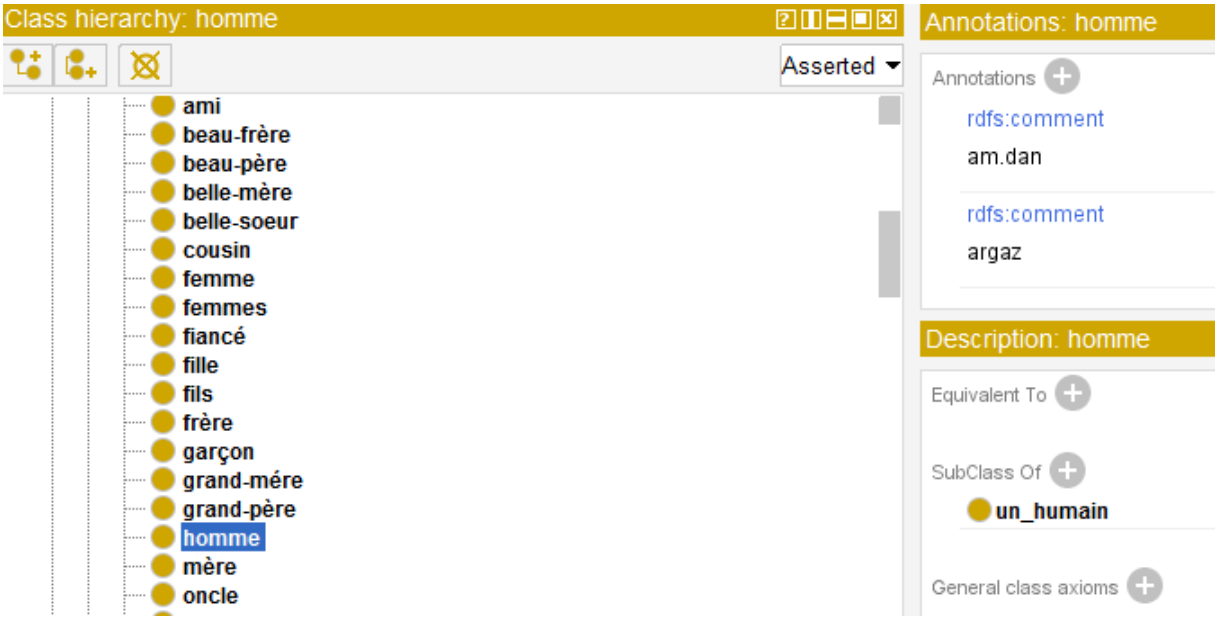

Figure 10 : Annotation « mot homme ».

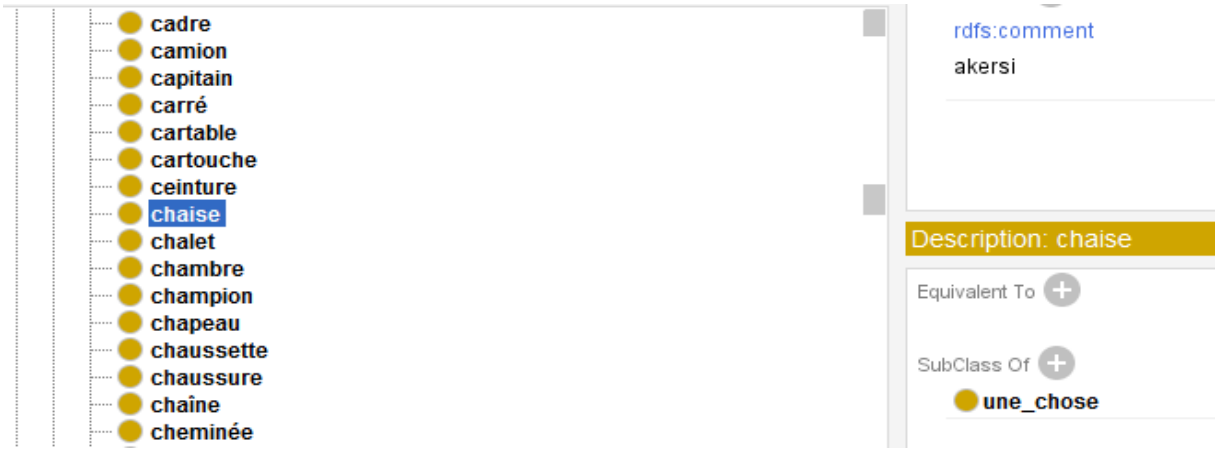

Figure 11 : Annotation « mot chaise ».

3.2. SPARQL

SPARQL est un langage de requête et un protocole qui permet de rechercher, d'ajouter, de modifier ou de supprimer des données RDF. Son nom représente Protocol and RDF Query Language.

SPARQL est considéré dès 2007 comme l'une des technologies clés du Web sémantique par Tim Berners-Lee l'inventeur du Web sémantique qui explique « *Tenter d'utiliser le Web sémantique sans SPARQL revient à exploiter une base de données relationnelles sans SQL* ».

Aujourd'hui, le Web des données est constitué de centaines de service SPARQL qui mettent à disposition de plus en plus de données au travers d'Internet comme le fait le projet Wikidata.[25]

3.2.1 Exemple du requête

Dans la figure 12 nous présentons une méthode qui s'appelle Trad_mot(verfmot). Elle prend en paramètre une chaine de caractère (le mot) pour vérifier que le mot existe dans la base de donnés.

```
private static String Trad_mot(String verfmot){
  String s="";
   OntModel model = OpenOWL.OpenConnectOWL();
   String queryString =
            "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                    "PREFIX owl:<http://www.w3.org/2002/07/owl#>" +
                    "PREFIX about:<http://www.semanticweb.org/chawki/ontologies/2020/5/francais-ontology-4#>" +
                    "SELECT DISTINCT * "+"WHERE {" +
                    verfmot +"rdfs:comment ?y." +
                    "}";
   Query query = QueryFactory.create(queryString);
   QueryExecution qexec = QueryExecutionFactory.create(query, model ) ;
   try {
       ResultSet results = qexec.execSelect() ;
       for ( ; results.hasNext() ; ) {
           QuerySolution soln = results.next() ;
           Literal y = soln.getLiteral("y") ;
           s=s+y.toString()+", ";
       }
   }finally {
       qexec.close();
       return s;
   }}
```

Figure 12 : Exemple de requête pour récupérer un mot dans l'ontologie.

Dans la figure 13 nous présentons une autre requête sparql qui sert à vérifier que l'argument passé en paramètre est vraiment un verbe ou non.

```
String queryString =
        "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                "PREFIX owl:<http://www.w3.org/2002/07/owl#>" +
                "PREFIX about:<http://www.semanticweb.org/chawki/ontologies/2020/5/francais-ontology-4#>" +
                "SELECT DISTINCT * "+"WHERE {" +
                "?x rdfs:subClassOf about:verbe." +
                "}";
```

Figure 13 : Exemple requête sparql pour vérifier que le mot est un verbe.

3.3 Conclusion

Ce chapitre traite le sujet du l'implémentation de l'ontologie avec l'outil Protégé. Nous avons présenté les relations entre les classes et finalement des exemples requêtes SPARQL pour interroger l'ontologie.

Chapitre 4 : Réalisation du traducteur

Chapitre 4 : Réalisation du traducteur

4.1. Introduction

L'objectif de ce chapitre est de concevoir et de réaliser un traducteur basé sur une ontologie de la langue française.

L'application suit une architecture client / serveur, grâce à des communications avec sockets.

Dans le coté serveur on implémenter les algorithmes de traduction français tamazight. Le côté client est une simple interface utilisateur.

Nous présentons tout d'abord, certains outils de développement que nous avons utilisés. Ensuite nous présentons les diagrammes UML avec les interfaces utilisateur et l'explication des algorithmes utilisés.

4.2. La démarche suivie

La conception de notre application est faite en utilisant le formalisme UML (Unified Modeling Language), le logiciel StarUML sera utilisé pour la conception.

Nous commençons par l'analyse fonctionnelle de notre système en déterminant les besoins des utilisateurs et les cas d'utilisation de notre système. Les interactions de l'utilisateur avec le système pour un tel cas d'utilisation sont décrites et schématisées par les diagrammes d'interactions en montrant leurs chronologies et scénarios de déroulement.

4.2.1. Architecture de l'application

L'application se compose d'une ontologie française qui se trouvent dans l'ordinateur, et une interface légère dans un smartphone. Un échange d'information (Socket) se déroulera comme suit.

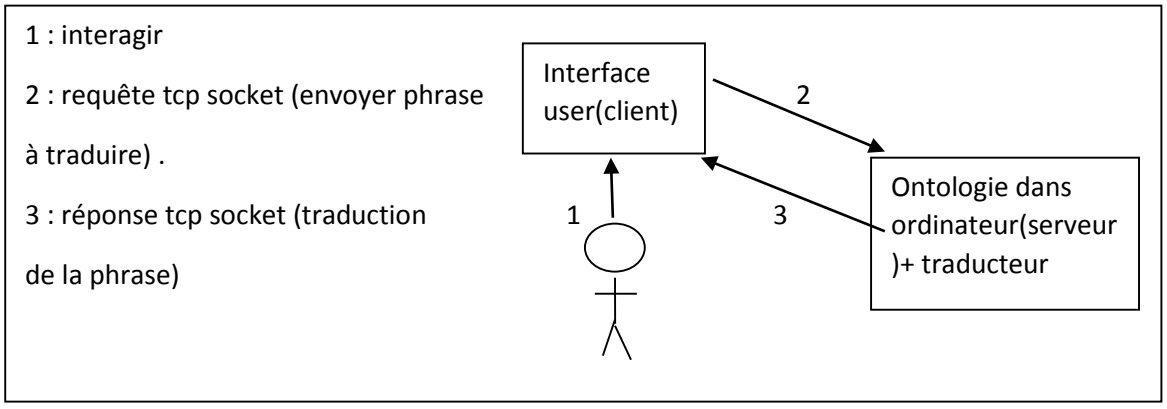

Figure 14 : Architecture de traducteur.

4.2.2. Diagramme de cas d'utilisation

Le diagramme des cas d'utilisation représente les principales fonctionnalités offertes par le logiciel pour chaque acteur. Ce diagramme donne un bon aperçu du système. C'est le premier diagramme du modèle UML.[26]

Identification des acteurs

- **Acteur principal** : c'est l'utilisateur de notre application.
- **Acteur secondaire** : c'est l'ordinateur où se trouve l'ontologie.

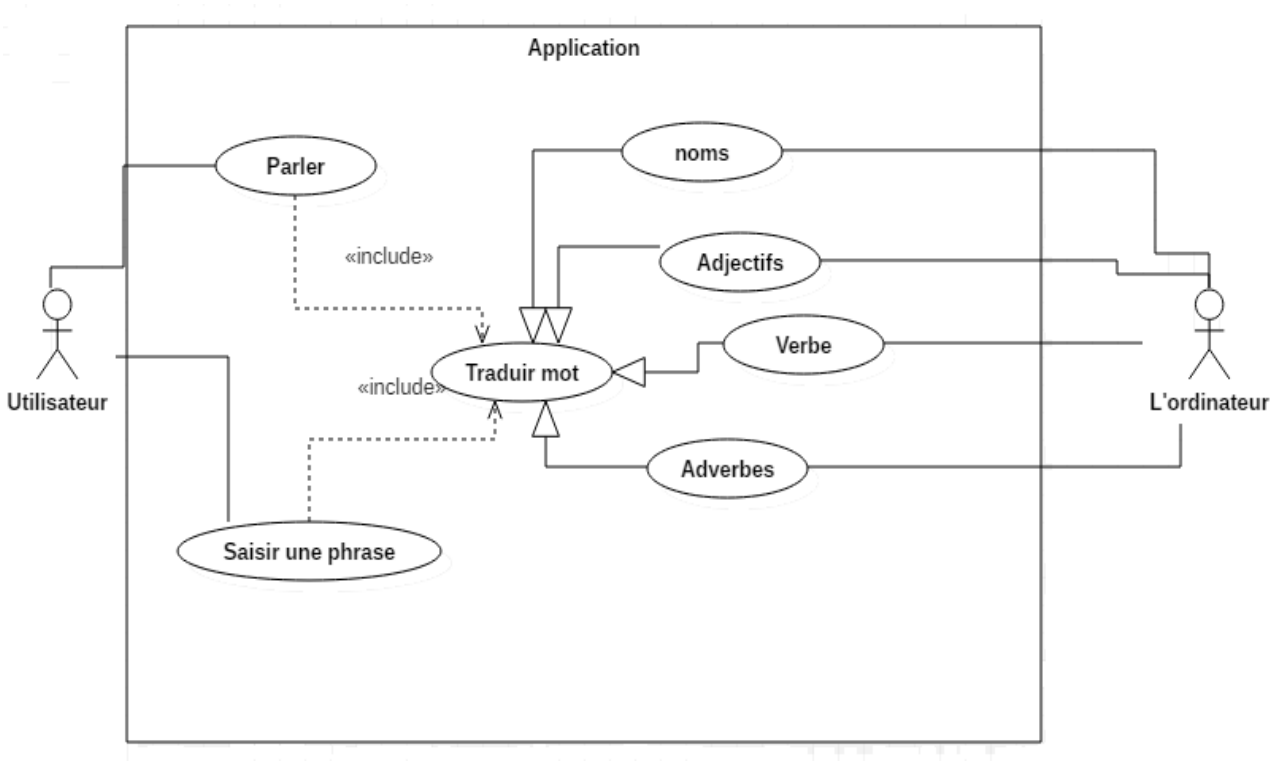

Figure 15 : Diagramme de cas d'utilisation.

4.2.3. Diagramme d'interaction

C'est une représentation d'un ensemble d'objets de classes différentes collaborant dans le cadre d'un scénario d'exécution du système. Dans ce diagramme, les objets s'envoient des messages qui invoquent des opérations sur les objets récepteurs. Le diagramme permet de suivre les interactions dynamiques entre objets et les traitements réalisés par chacun. [26]

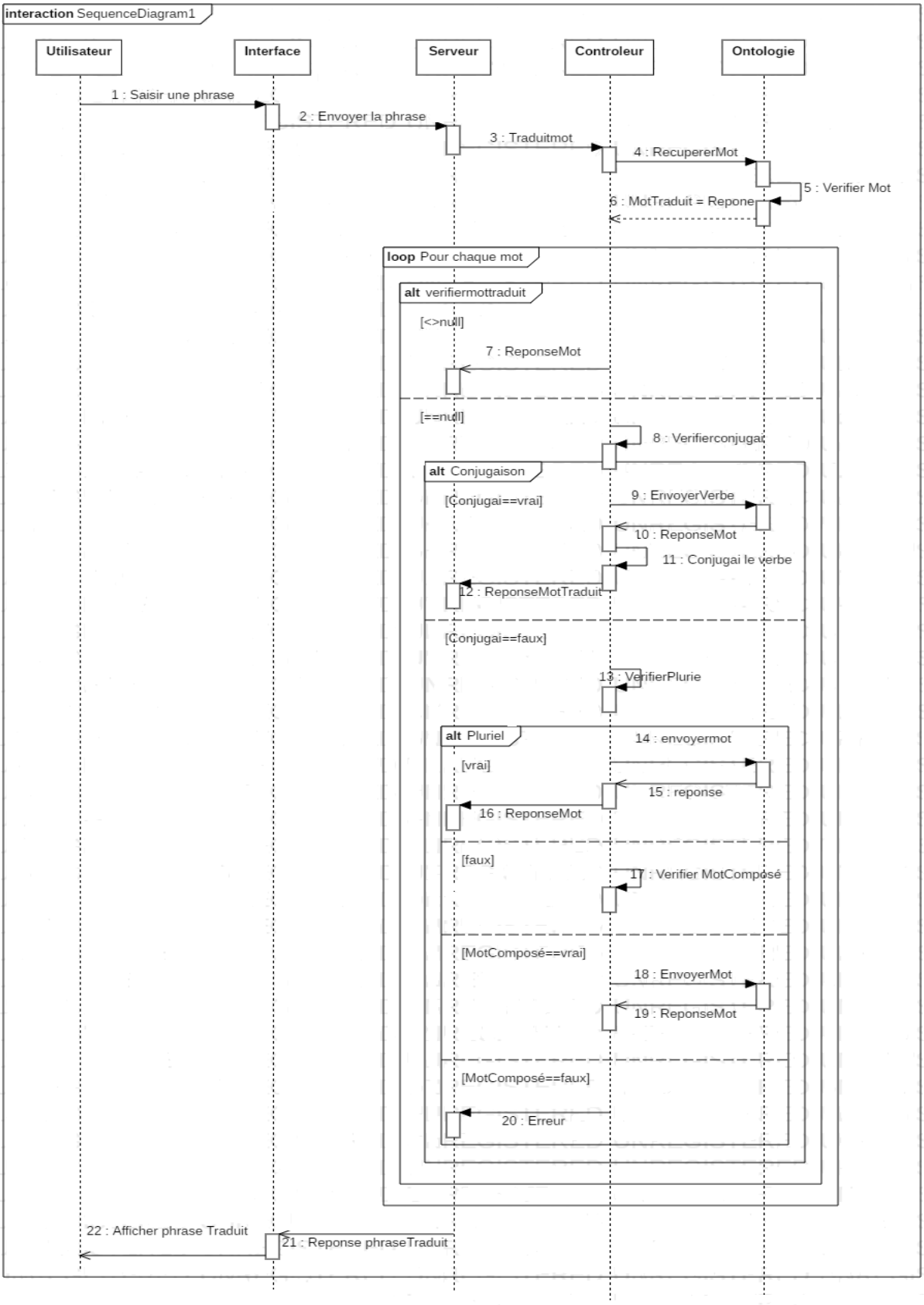

Figure 16: diagramme d'interaction.

4.2.4. Description textuelle

Nom du cas d'utilisation	Traduction d'une phrase simple
Objectif	Traduire une phrase du français au tamazight
Acteur	Utilisateur
Pré condition	Introduire une phrase par l'utilisateur
Scénario nominale	1-pour chaque mot(i) dans la phrase introduite 2-vérifier l'existence de mot(i) dans l'ontologie 3- récupérer sa traduction dans l'ontologie 4-afficher le résultat pour l'utilisateur si (TraduirMot<>null)
Scénario alternatif 1	Si (TraduirMot =null) : le mot(i) n'existe pas dans l'ontologie 1-vérifier qu'il est conjugué 2-transformer en infinitif 3-récupérer sa traduction dans l'ontologie 4-conjuguer le verbe récupéré 5-afficher le résultat pour l'utilisateur
Scénario alternatif 2	Si (TraduirMot =null) : le mot(i) n'existe pas dans l'ontologie 1-vérifier qu'il est au pluriel 2-transformer en singulier 3-récupérer sa traduction dans l'ontologie 4-mettre au pluriel mot récupéré 5-afficher le résultat pour l'utilisateur
Scénario alternatif 3	Si (TraduirMot =null) : le mot(i) n'existe pas dans l'ontologie 1-vérifier qu'il est un mot composé 2-décomposé le mot composé 3-récupérer leur traduction dans l'ontologie 4-afficher le résultat pour l'utilisateur
Scénario alternatif 4	Si (TraduirMot=null) : le mot(i) n'existe pas dans l'ontologie : La traduction du mot(i) = '' ''

Tableau 21 : La description textuelle.

4.3. Outils de développement

Nous avons des EDI pour le développement d'applications Android :« un EDI (Environnement de développement Intégré) est un logiciel qui regroupe un ensemble d'outils permettent de développer plus facilement.

4.3.1. Androïd Studio

C'est l'environnement de développement officiel pour le développement d'applications Androïd. Créé par Google et publié en 2014, il a remplacé les outils de développement d'Eclipse Il est disponible gratuitement sous licence Apache. Parmi les fonctionnalités principales d'Android studio nous avons :

- Une exécution instantanée des modifications du code dans l'application en cours d'exécution,
- Un émulateur riche qui permet de simuler des applications pour des appareils mobiles ou même des smart TV. Firebase Messaging, et Google End points pour l'intégration au cloud [27].

4.3.2. Jena (framework)

Apache Jena (ou juste **Jena**) est un framework Java gratuit et open source pour la construction des applications du web sémantique et de l'annotation sémantique (Linked data),Il servira à lire et interagir avec ontologie et traiter les données RDF [28].

4.4. Socket

Un socket est un point de terminaison d'une communication bidirectionnelle, c'est-à-dire entre un client et un serveur en cours d'exécution sur un réseau donné. Les deux sont liés par un même numéro de port TCP(TCP Layer) de sorte que la couche puisse identifier la demande de partage des données.[29]

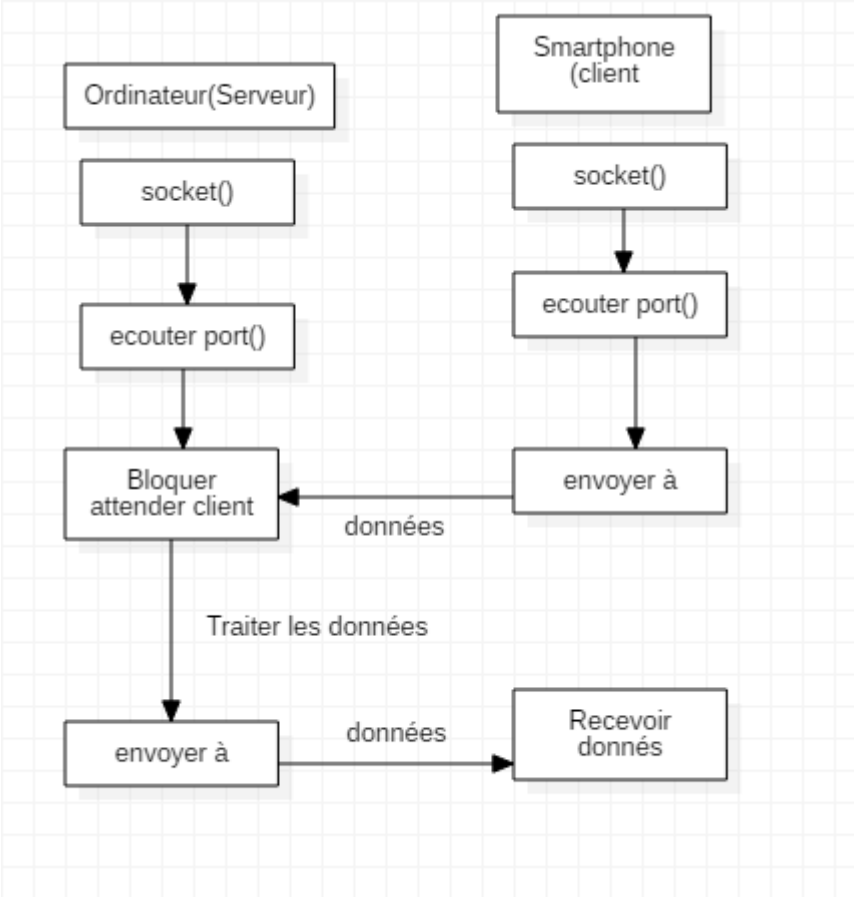

Figure 17 : Fonctionnement des sockets.

Dans la figure 17, nous avons un exemple de communication socket entre un ordinateur et le smartphone. Pour débuter l'échange des données il faut que les appareils écoutent sur le même port.

4.5 Les Interface de notre application

Nous présentons quelques exemples de la traduction des phrases simples. Ainsi que l'algorithme associé pour cette tâche.

4.5.1 Voice Speech to text

Nous avons utilisé la fonctionnalité de Google Voice speech recognition français pour la parole.

.

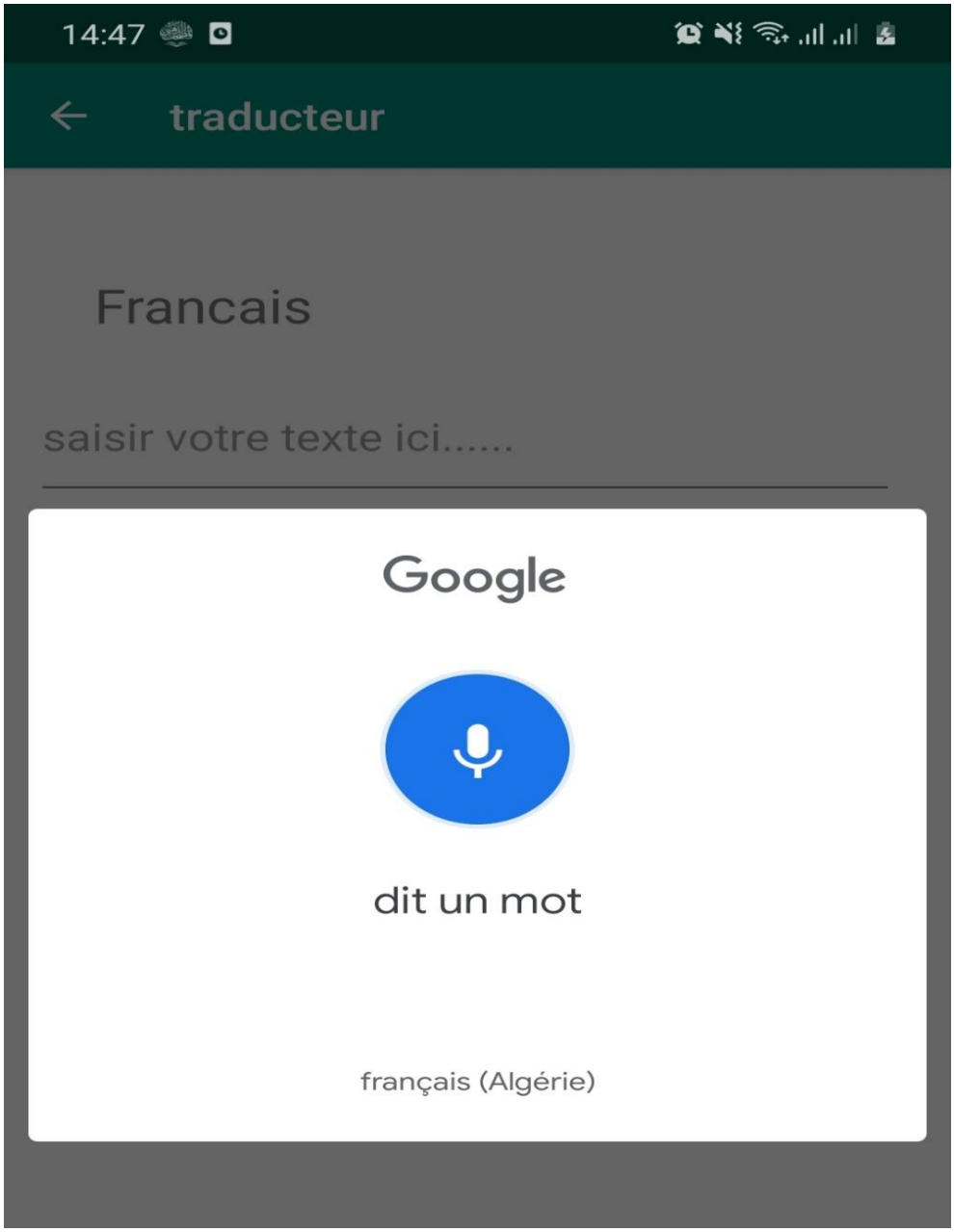

Figure 18 : Fonctionnalité speech recognition.

4.5.2 Traduction des mots

4.5.2.1 Pluriel / Féminin

Pour traduire un mot au pluriel, nous avons mis un algorithme pour détecter la forme de pluriel, ensuit transformer à la forme de singulier, pour à la fin le retransformer en pluriel dans la langue cible, la même chose pour le féminin.

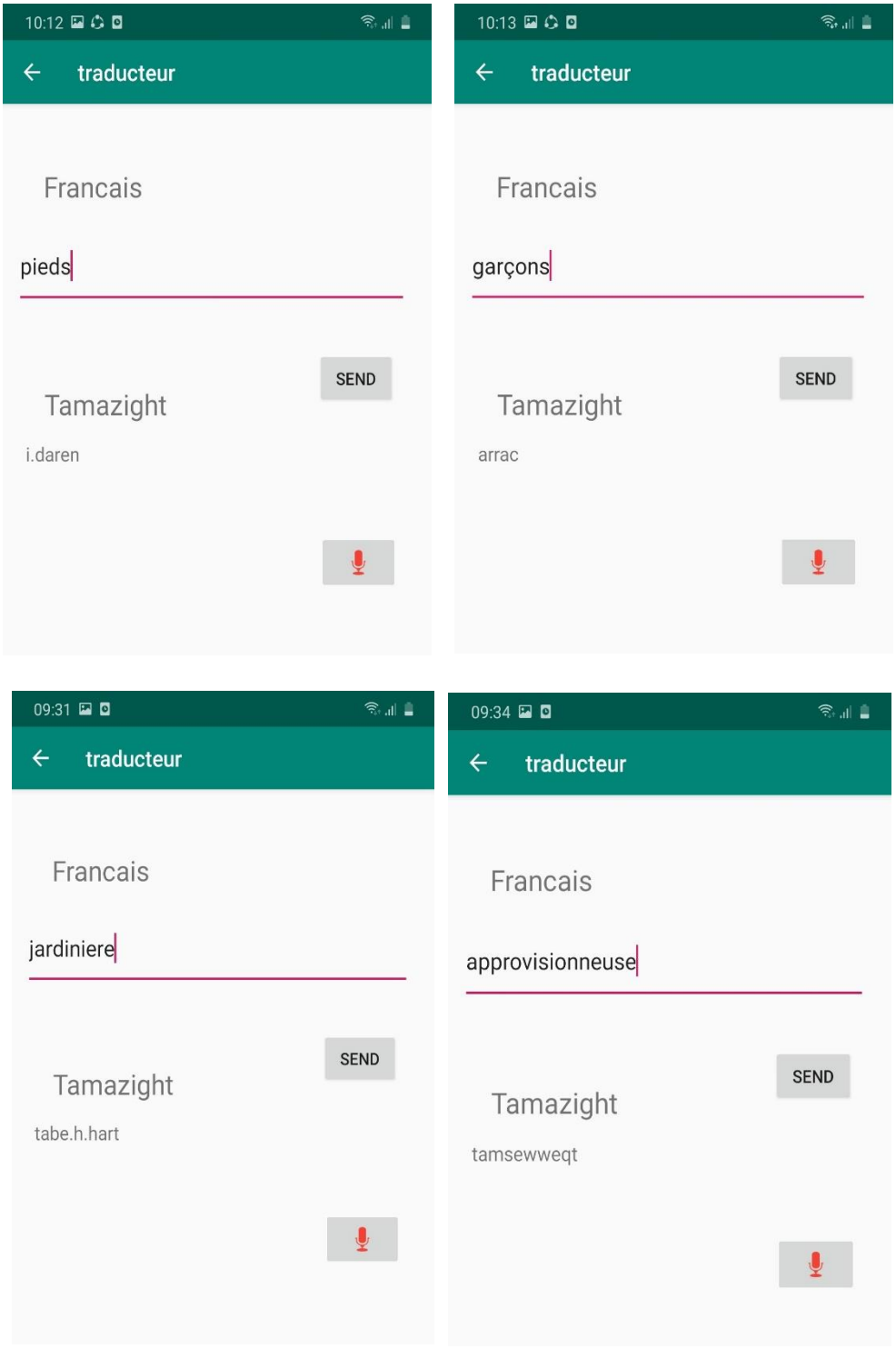

Figure 19 : *Exemples de la traduction des mots.*

4.5.2.2 Algorithme pour le féminin

Nous présentons le code source programmé afin de traiter les mots sous forme féminin.

Dans la figure 20 nous avons programmé un algorithme simple qui nous retourne f si le mot est féminin sinon rien grâce à la terminaison de mot.

```
 private static String Rfn (String motENT){
String r="";
  if(motENT.endsWith("e")||
          motENT.endsWith("ère") ||motENT.endsWith("nne")||motENT.endsWith("trice")||motENT.endsWith("euse")){
      r="f";
      return r;
    }

  return r;
}
```

Figure 20 : *Algorithme pour vérifier la forme féminine.*

La figure 21 nous montre un algorithme qui traite un mot Français féminin après la vérification, l'algorithme cherchera la forme masculine du mot en Tamazight dans l'ontologie si le masculin existe, le transformer en féminin.

```
 //traiter le mot feminin
vv=Rfn(motENT);
if(vv=="f"&& conjugai==false && mot_composer==false && pluriel ==false){
    String ff,f1,f2,ftamzi;
    ftamzi="";
    feminin =true;
    // trouver le nom masculin tamzight
    if(motENT.endsWith("e")){   motENT=motENT.substring(0,num-1);
        f1=motENT;
        f2 = "about:" + f1 + " ";
        ftamzi = Trad_mot(f2); } else  if(motENT.endsWith("ère")){ motENT=motENT.replaceAll("ère","er");
        f1=motENT;
       f2 = "about:" + f1 + " ";
        ftamzi = Trad_mot(f2); }else if(motENT.endsWith("euse")){ motENT=motENT.replaceAll("euse","eur");
        f1=motENT;
       f2 = "about:" + f1 + " ";
        ftamzi = Trad_mot(f2);   }else if(motENT.endsWith("trice")){motENT=motENT.replaceAll("trice","eur");
        f1=motENT;
        f2 = "about:" + f1 + " ";
        ftamzi = Trad_mot(f2);  }else if(motENT.endsWith("nne")){  motENT=motENT.replaceAll("nne","on");
        f1=motENT;
        f2 = "about:" + f1 + " ";
        ftamzi = Trad_mot(f2); }
    // si le masculin est trouvé alors le noms tamazight est transformé en féminin
    if(ftamzi!=""){   String[] fff =ftamzi.split(",");
        System.out.println(fff[0]);
       String fem="t"+fff[0]+"t";
        System.out.println(fem);
        ph_trad=ph_trad+fem;  }
}
```


Figure 21 : Algorithme pour le féminin.

4.5.2.3 Algorithme pour le pluriel

Nous présentons le code source programmé afin de traiter les mots au pluriel.

La figure 22 présente un algorithme qui traitre les mots Français au pluriel. Il cherchera la forme singulière de mot en Tamazight ensuite le transformer au pluriel avec la méthode plu_A_EN() qui est présentée dans la figure suivante 23.

```
//traiter pluriel
    //verifier que le mot est au puriel
String vv=Rpn(motENT);
                            if(vv=="plu" && conjugai==false && feminin==false ) {
                              String p,plu1,plu2;
                             pluriel =true;
                              String plutamzi="";
                              //récupérer le mot en tamazight
                                  if(motENT.endsWith("aux")){
                                      motENT=motENT.replaceAll("aux","al");
                                      plu1=motENT;
                                      plu2 = "about:" + plu1 + " ";
                                      plutamzi = Trad_mot(plu2);
                                  }else
                                  if(motENT.endsWith("s")|| motENT.endsWith("x")){
                                 motENT=motENT.substring(0,num-1);
                                 plu1=motENT;
                                 plu2 = "about:" + plu1 + " ";
                                 plutamzi = Trad_mot(plu2);
                              }//si le mot existe transformer le en pluriel
                              if(plutamzi!=""){
                                  plurieltam plu = new plurieltam();
                                  String[] pp =plutamzi.split(",");
                                  System.out.println(pp[0]);
                                  p=plu.plur_A_EN(pp[0]);
                                  System.out.println(p);
                                  ph_trad=ph_trad+p;
                              }
                            }
```

Figure 22 : Algorithme pour le traiter le mot en entré pluriel.

```
public class plurieltam {
    String plu_UNC[]={};
    String plu_INC[]={};
    // les pluriel iregulier
    String plu_IR[]={"aqcic","arrac",};
    String plu_I[]={};
    String plu_AT[]={};
    //d'aprés les régle de pluriel tamazight // manque d'autre cas du pluriel com AT ...
    public String plur_A_EN(String m){
int d=m.length();
        String mot=m;
        for(int i = 0; i < plu_IR.length; i++){
            if(mot.equals(plu_IR[i])){
                mot=plu_IR[i+1];
                return mot;
            }}
        if (mot.endsWith("i") && mot.startsWith("a"))
        {mot=mot.replaceFirst("a","i");
            mot=mot.substring(0,d-1);
            mot=mot+"yen";
            return mot;
        }else  if (mot.startsWith("a")){
            mot=mot.replaceFirst("a","i")+"en";
            return mot;

        }else if(mot.startsWith("u")){
            mot=m.replaceFirst("u","i");
            return mot;
        }else mot=mot+"en";
        return mot;    }}
```

Figure 23 : Algorithme pour transformer le mot Tamazight au pluriel.

4.5.2.4 Algorithme pour les mots composés

Nous présentons le code source programmé afin de traiter les mots composés.

```
    //traiter mot composé
    if(motENT.contains("-")){
        // couper le mot composé en deux mot
        motcompo=motENT.split("-");
        mot_composer=true;
        motENT=motcompo[0];
        awsil=motcompo[1];
        System.out.println(awsil);
        if(mot_composer==true){
            //chercher les mots dans la base de connaisance
            String mc1="about:" + motENT + " ";
            String mc2="about:" + awsil + " ";

            String m1=Trad_mot(mc1);
            String m2=Trad_mot(mc2);
            ph_trad=ph_trad+m1+" "+m2;
            ph_trad=ph_trad.replaceAll(","," ");
        }

    }
```

Figure 24 : Algorithme pour le mot composé.

4.5.3 Traduction des verbes

Pour traduire un verbe conjugué, nous avons mis un algorithme pour détecter le temps grâce à sa terminaison, ensuit le transformer à la forme infinitive, pour qu'à la fin, le conjuguer dans la langue cible.

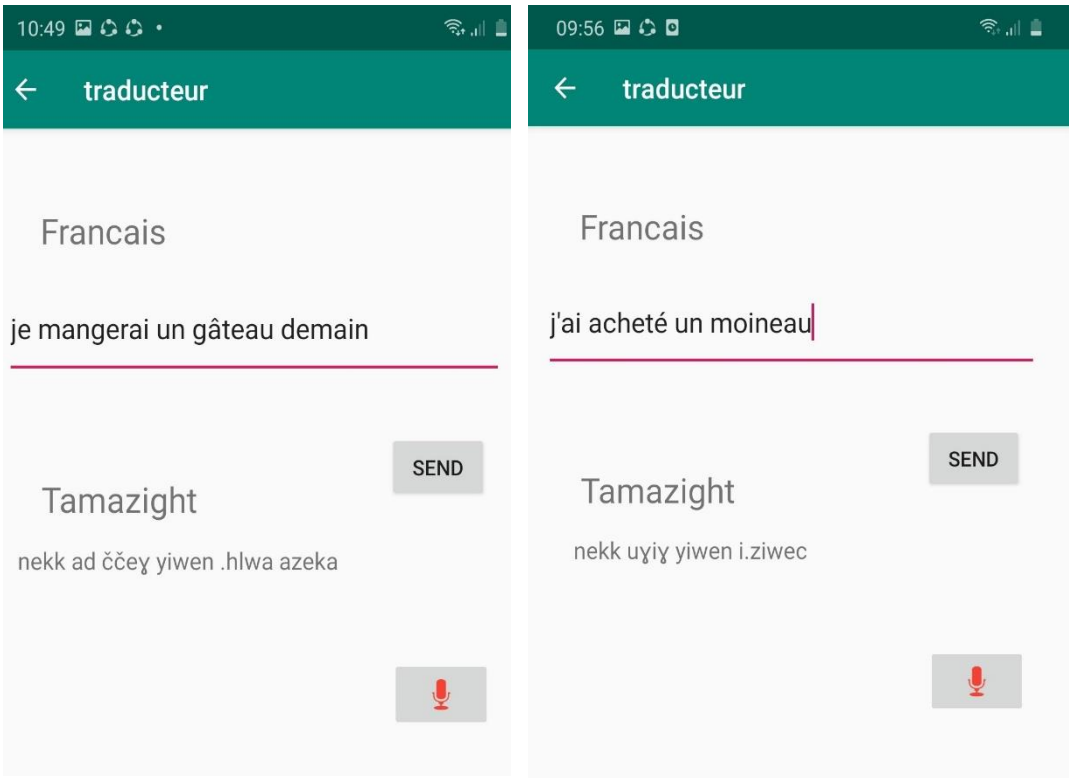

Figure 25 : Exemple de la traduction des verbes.

4.5.3.1 Algorithme pour le présent :

Nous présentons le code source programmé afin de traiter les verbes conjugués au présent.

Dans les figures 26, 27 et 28 présentent des exemples de conjugaison de prénom je, nous et tu. Des verbes au présent après avoir récupérer leur infinitif en regardant la terminaison de verbe.

```
//conjugai présent
 if(verbeTemps=="pr" && tamzi !=""){conjugai=true;
     ve[0] =tamzi.replaceAll(",","");

     if (motENTR[i-1].equals("je") ) {
         System.out.println(d);
         //doubler les lettres pour donner importance de l'action
         if(tamzi.contains("čč") ) {
             tamzi = tamzi.replaceAll("čč", "tt");
             ;
         }
         if(tamzi.contains("r") ){
             tamzi=tamzi.replaceAll("r","rr");
         }
         if(tamzi.startsWith("e")){
             ph_trad=ph_trad+"te"+tamzi.substring(1,d-2)+"eɣ";
         }
         else
             ph_trad=ph_trad+"tt"+ve[0].substring(0, d-2)+"eɣ";
     }
```

Figure 26 : Algorithme conjugaison prénom « je » en présent.

```
else if (motENTR[i - 1].equals("nous")) {
    if(tamzi.contains("čč") ) {

        tamzi = tamzi.replaceAll("čč", "tt");

    }
    if(tamzi.contains("r") ){

        tamzi=tamzi.replaceAll("r","rr");
    }if(tamzi.startsWith("ett")){

        ph_trad=ph_trad+"n"+tamzi.substring(1,d-2)+"et";

    } else
    if(tamzi.startsWith("e")){

        ph_trad=ph_trad+"n"+tamzi.substring(1,d-2);

    }
    else
        ph_trad=ph_trad+"nett"+ve[0].substring(0, d-2);
}
```

Figure 27 : Algorithme conjugaison prénom « nous » en présent.

```
else if (motENTR[i-1].equals("tu")) {
    if(tamzi.contains("čč") ) {
        tamzi = tamzi.replaceAll("čč", "tt");
    }
    if(tamzi.contains("r") ){
        tamzi=tamzi.replaceAll("r","rr");
    }
    if(tamzi.startsWith("e")){
        ph_trad=ph_trad+"te"+tamzi.substring(1,d-2)+"eḍ";

    } else
        ph_trad=ph_trad+"tett"+ve[0].substring(0, d-2)+"eḍ";
}
```

Figure 28 : Algorithme conjugaison prénom « tu » en présent.

4.5.3.2 Algorithme pour le passé composé :

Nous présentons le code source programmé afin de traiter les verbes conjugués au passé composé.

Les figures 29 et 30 présentent des exemples pour traiter les verbes conjugués au passé composé de prénom « je, il ».

```
      //conjugai passé composé
if(verbeTemps=="pc" && tamzi!="" ){
    conjugai=true;

    ve[0] =tamzi.replaceAll(",","");

    if (motENTR[i-1].equals("ai")  && motENTR[i-2].equals("j")) {

        if(tamzi.startsWith("e") && d<=5){

            ph_trad=ph_trad+" nekk "+tamzi.substring(1,d-2)+"iɣ";
        }else if (tamzi.startsWith("a")&& d<=5){

            ph_trad=ph_trad+" nekk "+"u"+tamzi.substring(1,d-2)+"iɣ";}

        else if(tamzi.endsWith("r") && d>5){
            ph_trad=ph_trad+" nekk "+ve[0].substring(0, d-2)+"eɣ";
        }else if(d<5){
            ph_trad=ph_trad+" nekk "+ve[0].substring(0, d-2)+"iɣ";
        }else
            ph_trad=ph_trad+" nekk "+ve[0].substring(0, d-2)+"eɣ";

    }
```

Figure 29 : Algorithme conjugaison prénom « je » en passé composé.

```
else if (motENTR[i-2].equals("il") && motENTR[i-1].equals("a")) {
    if (tamzi.startsWith("e") && d <= 5) {
        ph_trad = ph_trad + "ye" + tamzi.substring(1, d - 2) + "a";
    } else if (tamzi.startsWith("a") && d <= 5) {

        ph_trad = ph_trad + "yu" + tamzi.substring(1, d - 2) + "a";
    } else if ( d <= 5) {

        ph_trad = ph_trad + "i" + tamzi.substring(1, d - 2) + "a";
    }else   if (tamzi.startsWith("a") && d > 5) {

        ph_trad = ph_trad + "yu" + tamzi.substring(1, d - 2) ;
    } else if (tamzi.endsWith("r") && d > 5) {
        ph_trad = ph_trad + "y" + ve[0].substring(0, d - 2) ;
    } else {
        ph_trad = ph_trad + "y" + ve[0].substring(0, d - 2) ;
    }
}
```

Figure 30 : Algorithme conjugaison prénom « il » en passé composé.

4.5.3.3 Algorithme pour future :

Nous présentons le code source programmé afin de traiter les verbes conjugués au future. La figure 31 montre l'algorithme responsable pour conjuguer les verbes Tamazight au future.

```
                    int d =tamzi.length();
//conjugai future
                        if( ( verbeTemps=="f") && (tamzi !="" ) ) {
conjugai=true;
       ve[0] =tamzi.replaceAll(",","");
     if (motENTR[i-1].equals("je") ) {
                                    if(tamzi.startsWith("e")){
                  p                 h_trad=ph_trad+"ad "+tamzi.substring(1,d-2)+"eɣ";
                                    }
                                    else   ph_trad=ph_trad+"ad "+ve[0].substring(0, d-2)+"eɣ"; }
     else if (motENTR[i-1].equals("tu")) {  if (tamzi.startsWith("e")) {
                                             ph_trad = ph_trad + "a " + "te" + tamzi.substring(1, d-2)+"eḍ";
                                         } else
                                             ph_trad = ph_trad + "a " +"t" +ve[0].substring(0, d-2) + "eḍ"; }
    else if (motENTR[i - 1].equals("il")) {      if (tamzi.startsWith("e")) {
                                              ph_trad = ph_trad + "ad " + "ye" + tamzi.substring(1, d-2) ;
                                         } else
                                          ph_trad = ph_trad + "ad " + "ye"+ve[0].substring(0, d-2) ;
                                           }
    else if (motENTR[i - 1].equals("vous")) {if (tamzi.startsWith("e")) {
                                               ph_trad = ph_trad + "a " + "te" + tamzi.substring(1, d-2)+"em" ;
                                               } else
                                             ph_trad = ph_trad + "a " + "t"+ve[0].substring(0, d-2)+"em" ;}
    else if (motENTR[i - 1].equals("nous")) {if (tamzi.startsWith("e")) {
                                              ph_trad = ph_trad + "a " + "ne" + tamzi.substring(1,d-2) ;
                                             } else
                                            ph_trad = ph_trad + "a " + "ne"+ve[0] .substring(0, d-2);  }
    else if (motENTR[i - 1].equals("ils")) {if (tamzi.startsWith("e")) {
            ph_trad = ph_trad + "ad "  + tamzi.substring(1, d-2)+"en" ;
        } else
            ph_trad = ph_trad + "ad " +ve[0].substring(0, d-2)+"en" ;
```

Figure 31 : Algorithme conjugaison future.

4.5.4 Traduction des phrases :

Pour traduire une phrase, nous avons mis un algorithme qui traite chaque mot, pour ensuite afficher le résultat.

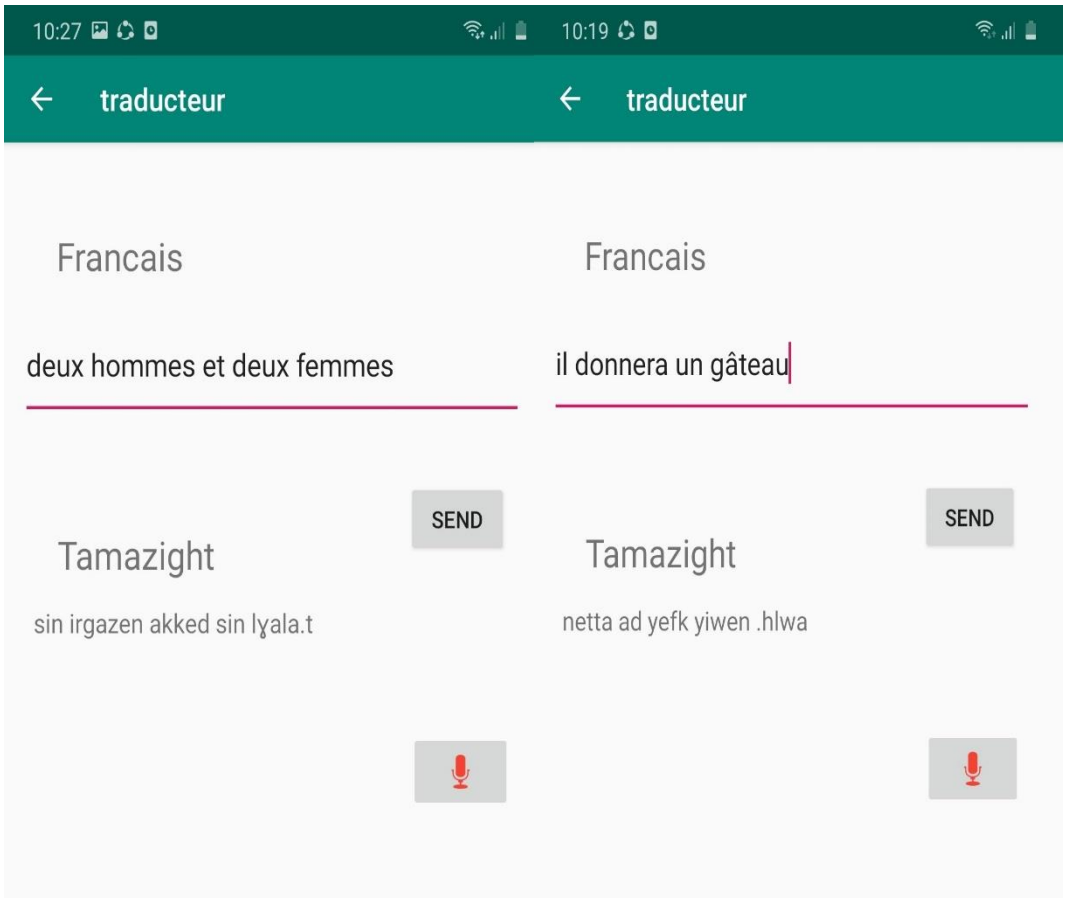

Figure 32 : Exemple de la traduction des phrases simple.

4.6. Conclusion

Dans ce chapitre, nous avons présenté le cadre technique de réalisation de notre traducteur Français – Tamazight. L'élaboration d'une ontologie est une tâche complexe et difficile. nous avons pu élaborer une petite ontologie qui peut au moins aider à donner un exemple pour la réalisation d'un traducteur pour notre langue maternelle.

Conclusion et perspectives

Conclusion et perspectives

Le but principal de ce projet était de concevoir un traducteur bidirectionnel (tamazight-français) qui fonctionne sous Android et soit basé sur l'ontologie.

Ce travail commence avec la création de l'ontologie des mots français (langue source) et leurs significations en tamazight (langue cible). Ensuite nous avons étudié la linguistique et la grammaire des deux langues. Nous avons développé des algorithmes de traduction tels que (les verbes conjugués au (future, passé simple, présent, passé composé), les mots au pluriel et les mots composés). Nous avons implémenté un traducteur suivant l'architecture client/serveur, les communications se font avec des sockets.

Au cours de ce travail nous nous sommes rendus compte qu'il y a toujours un manque de référence (absence de sources de données, et seulement quelques publications). D'autres problèmes sont liés à la création de nouveaux contenus. Absence de certains alphabets comme (ɣ, ḥ, ṛ) dans le logiciel Protégé.

Nous avons appris que la langue tamazight est riche dans ses concepts et sa grammaire, aussi que la langue française. Nous avons amélioré notre maitrise dans le logiciel Protégé et ses relations. Nous avons donc appris la définition d'ontologie et ses domaines de pratique.

Les perspectives de nos travaux sont multiples :

Notre ontologie a besoin d'être enrichie avec plus de concepts et de relations sémantiques.

Nous avons utilisé speech recognition de Google pour la langue française, et il faudra encore trouver et développer une reconnaissance vocale pour la langue tamazight.

Nous n'avons pas pu créer les algorithmes dont on a besoin pour une traduction complète et détaillée comme par exemple la forme passive et active, la forme négative et interrogative, et des algorithmes pour reconnaitre des phrases plus complexes.

Nous souhaitons rendre notre traducteur compatible avec plusieurs plateformes mobiles et qu'il devienne multidirectionnel avec d'autres langues anglais, arabe, etc.

Ce travail a été très intéressant à développer, nous somme intéressé à faire une version simple et légère. Elle peut être améliorée avec d'autre fonctionnalités. C'est pourquoi nous restons ouverts à toutes les critiques et sommes prêts à recevoir toutes les suggestions et remarques afin d'améliorer d'avantage ce travail.

Bibliographie

Bibliographie :

[1].https://linguistique-amazighe.blog4ever.com/la-traduction-vers-le-berbere-de-kabylie-etatdes-lieux-et-critiques-2005(Consulté 2021/06/20).

[2]. S.Bessaoudi , H,Chilla, Conception et réalisation d'un traducteur Tamazight –Français sous Android, Mémoire de master,Université de bejaia 2020.

[3]. Gongres mondial amazigh, Comité des Droits de l'Homme des Nations Unie 90° session, Genève, 9-27 juillet 2007

[4]. Ameur M., Bouhjar A., Boukhris F., Boukous A., Boumalk A., Elmedlaoui M., Iazzi E. et Souifi H, Initiation à la langue amazighe, Maroc : IRCAM, 2004.

[5]. Alphabet berbère latin — Wikipédia (wikipedia.org) (Consulté 2021/05/29).

[6]. https://www.aclweb.org/anthology/F13-1001.pdf(Consulté 2021/05/29).

[7]. Michel Quitout,Grammaire berbere (rifain, tamazight, chleuh), L'Harmattan, Paris 1997.

[8]. Salem Chaker,Linguistique berbère: études de syntaxe et de diachronie, Peeters Publishers, 1995.

[9]. A. Hanoteau, Essai de grammaire kabyle, Bastide, 1858.

[10]. https://fr.wikipedia.org/wiki/Francais, (Consulté le 25/06/2021).

[11]. https://www.francaisfacile.com/exercices/exercice-francais-2/exercice francais84631.php. (Consulté le 25/06/2021).

[12]. Y. Delatour, D. Jennepin, M. Léon-Dufour, B. Teyssier, Nouvelle grammaire du français,1991, 367 p.

[13]. T.Berners-Lee, J. Hendler et O.Lasilla, The Semantic Web. Scientific American, 284 2001.

[14]. A. Rector. Thesauri and formal classifications : Terminologies for people and machines. Methods of Information in Medicine, 37(4_5), 501_509, 1998.

[15]. N. Aussenac-Gilles et A. Busnel. "Méthode de construction à partir du texte d'une ontologie du domaine de l'industrie de la fibre de verre". Rapport Interne IRIT/2002-11-R.

Avril 2002.

[16]. Fouzia Amourache. Construction d'une ontologie pour l'annotation des cvs/offres d'emplois. Mémoire de Magister. Université Mentouri de Constantine,2008.

[17]. Hacine Gherbi. Construction d'une Ontologie pour le WEB Sémantique. Mémoire de Magister, École doctorale d'informatique, Université de Setif, 2014.

[18]. Gomez Pérez A., Benjamins V.R. "Overview of Knowledge Sharing and Reuse Components : Ontologies and problem-Solving Methods". Proceeding og the IJCAI-99 workshop on Ontologies and problem-Solving Methods (KRR5), Stockholm (Suède), pp. 1.1-1.15 (1999).

[19]. M. Uschlod & M. Grüninger. "ONTOLOGIES : princples, Methods and applications ". Knowledge Engineering Review. 1996.

[20]. M. Fernandez, A. Gomez-Perez et N. Juristo. " METHONTOLOGY: from ontological art toward ontological engineering". Spring symposium series on ontological engineering. USA, 1997.

[21]. https://de.wikipedia.org/wiki/DAML%2BOIL (Consulté le 12 /10 /2021).

[22]. Ontology Web Language (OWL). http://www.w3.org/OWL/ (Consulté le 12/6/2021).

[23]. IDE protégé : https://www.protege.stanford.edu/ (Consulté le 14/06/2021).

[24]. https://fr.wikipedia.org/wiki/DeepL (Consulté le 12 /10 /2021).

[25]. https://fr.wikipedia.org/wiki/SPARQL (Consulté le 22 /10 /2021).

[26]. Pascal Roques, Uml 2 par la pratique études de cas et exercices corriges, 2009, 396 p.

[**27].** https://www.supinfo.com/articles/single/6135-top-10-ide-developpeurs-java. (Consulté le 01/7/2021).

[28]. https://fr.wikipedia.org/wiki/Jena_(framework) (Consulté le 01/07/2021).

[29]. https://devstory.net/10393/java-socket (Consulté le 24/08/2021).

Résumé

Ce travail vise à développer un traducteur de français (langue source) à tamazight (langue cible) sous Android, avec l'aide d'ontologies.

Nous avons continué un travail qui traite la traduction de tamazight au français. Pour cela nous avons formé une ontologie qui contient environ plus de 1300 mots. Ensuite, nous avons élaboré des algorithmes pour l'enrichissement de ce traducteur par des mots dérivés grâce à la grammaire de deux langue, nous avons aussi utilisé des sockets pour permettre la communication entre l'ordinateur où se situe le traducteur et le smartphone, qui est une simple interface utilisateur exploitant cette ontologie afin de traduire des mots et de phrases simples.

Mot clés : ontologie, Protégé OWL, Tamazight, Socket.

Abstract

This work aims to develop a translator from French (source language) to Tamazight (target language), based on ontology using android.

We continued a work that deals with the translation from Tamazight to French. For this we have formed an ontology that contains over 1300 words. Then, we developed algorithms for the enrichment of this translator by words derived from the grammar of the two languages, we also used sockets to allow communication between the computer where the translator is located and the smartphone, which is a simple user interface exploiting this ontology in order to translate simple words and sentences.

Key words: Ontology, Protégé OWL, Tamazight, Socket.