République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Département Informatique

Module : TALN

Rapport du mini-projet TALN

Réalisation par : **- CHABANE Nouar**
-BEZGALI Meriem
-CHENNIT Naila
-MOUSSAOUI Amina

M2 SII
Groupe : 01

I. Introduction :

« **Le traitement automatique des langues** (abr. **TAL**), est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle. Il vise à créer des outils de traitement de la langue naturelle pour diverses applications. Il ne doit pas être confondu avec la linguistique informatique, qui vise à comprendre les langues au moyen d'outils informatiques. Le TALN est sorti des laboratoires de recherche pour être progressivement mis en œuvre dans des applications informatiques nécessitant l'intégration du langage humain à la machine. » [1]

Problématique :

Dans ce mini-projet, il nous est demandé de développer un environnement qui permettra aux lexicographes de la langue Arabe de construire des dictionnaires de la langue sur des bases historiques, en mettant à leurs dispositions un corpus et l'outillage nécessaires. [2]

Plan et objectifs :

Le plan à suivit pour la réalisation de ce projet est comme suit :

- Recherche et extraction des textes arabe pour chaque période historique,
- Organisation et prétraitement du corpus,
- Fonctionnalité et code implémenté,
- Visualisation des résultats,

II. Recherche et extraction des données

La recherche de données s'est faite exclusivement sur le web. Nous disposions de contenus assez riches en format PDF mais nous n'avons pas réussi à faire la conversion nécessaire dû aux contraintes liées à l'encodage.

Tout d'abord nous avons repartis les corpus d'après les périodes historiques (basé sur Wikipédia). [3]

Une période historique est située dans un intervalle de temps et ne dépend nullement d'une région mais se distingue par le style d'écriture répandue au cours de celui-ci.

Afin de mener à bien la récolte de données nous avons utilisé l'outil **BeautifulSoup** qui est une librairie de python faisant du **scrapping** sur le web :

Fonction Extraction (Exemple Poeme) :

Entrée : url

Sortie : document

Début :

Extraire tous les auteurs disponibles sur le site à partir de l'url

Pour chaque auteur :
Ajouter la balise <author id= ‘’nom de l'auteur ‘’> au document
Extraire les poèmes de cet auteur
Pour chaque poème :
Id=1
Ajouter la balise <poeme_title id= ‘’titre du poème‘’> au document.
Pour chaque deux vers successifs :
Joindre les afin former un couple de vers.
Ajouter la balise <couple id =' +Id+'> ainsi que le couple de vers au document.
Fermer la balise <couple>.
Id+=1 ;
FinPour
Fermer la balise <poeme_title>
FinPour
Fermer la balise <author>
FinPour

Retourner document
Fin.

Nous avons obtenu suite à l'extraction les corpus suivants :

- **Corpus** Quran :

Sachant que ce texte est sacré et que son contenu est transcendant dans le temps, nous avons préféré le séparer du reste des textes. Ainsi que pour lui accorder la présentation appropriée.

Source : [4]

- **Corpus ère préislamique** (150 ans avant el hidjra) : Ce corpus est composé uniquement de poème en raison de l'absence d'autre type de texte après une longue recherche sur divers sites.

Source : [5] [6]

- **Corpus ère islamique** (0 hijri à 40hijri) **:**

Les données que nous avons récoltées touchent aux récits narratifs (nathr), aux poèmes (chiir) et aux hadiths du prophète (saaw). Ces derniers n'étant pas forcément correctes (sahih) vu que notre principale et unique source était contenue sur le web.

Etant donnée qu'il y avait beaucoup de textes qui se rapetissait dans un site ou un autre, nous avons choisi les sites qui englobait la plus grande majorité et qui disposait d'une meilleure qualité (complet).

Source : [5] [6] [7] [8]

- **Corpus ere Ummayad :**

Cette période date de l'an 40 hijri à l'an 132 hijri. Les données récoltées sont aussi variées que la période islamique.

Source : [5] [6] [9]

- **Corpus ère Beni Ababas** (132 hidjri à 656) :

Ce corpus contient deux styles le premier concerne uniquement les poèmes, tandis que le second concerne les textes narratifs récoltés. Les plus célèbres poètes de l'ère Bani Abbas (Abu al-Tayyib al-Mutanabi ..., etc.) de son célèbre poème «واحر قلباه ممن قلبه شبم ». Il dit :

الليل والخيل والبيداء تعرفني ** والرمح والسيف والقرطاس والقلم

Source : [5] [6] [8]

- **Corpus ère El douwel El moutatabia (**عصر الدول المتتابعة**)** :

Ce corpus englobe un ensemble de textes et de poèmes de la période Ottomane et Mamlouk en raison de la similarité du style d'écriture, cette période s'étant de 648 hijri à 1213.

Source : [5] [6] [7] [8]

- **Corpus ère moderne** (à partir du seizième siècle == 1213 hijri) :

Ce dernier comporte trois styles, les poèmes, les textes (extrait de livres) et les articles de journaux.

Source : [8] [9]

Il est à préciser que le site web [6] contient plus de 7000 livres, mais en raison de notre mauvaise connexion nous avons extrait plus de 150 livres de 100 pages pour chaque période, avec le nom et la date de mort de l'auteur pour classifier ensuite les livres automatiquement à l'aide d'un script.
Aussi nous avons été sélectives par rapport aux livres disponibles pour que notre dictionnaire historique n'ait pas de penchant envers telle ou telle école de pensée.

Ainsi que l'extraction des livres du site web [8] s'est faite grâce à un script (**Bot**).

Dans ce qui suit, se présente le pseudo-code de la classification des livres extraits d'après l'ère :

Fonction orderBooksShamela :

Entrée : chemin contenant les livres sous format *.txt à ordonnée (exemple : تلخيص الخطابة_ابن رشد الحفيد_ 595 .txt)

Sortie : organisation des fichiers dans leurs répertoires selon la période

Pour chaque fichier :

Spliter son nom selon le caractére « _ » dans une liste
Récupérer la date de mort de l'auteur
Vérifier dans quel intervalle il correspond, selon la date de mort de l'auteur, et l'affecte à un répertoire d'une phase

Fin.

III. Nettoyage des données :

Dans le but d'avoir une extraction lisible et correcte, nous avons procédé à un nettoyage des textes extrait, qui consiste en :

- L'élimination des chaines de caractère d'une autre langues étrangères à l'arabe, cela en utilisant le module **Pyarabic** de **Python**.
- L'élimination des **stopwords**, en arabe « المستبعادات » tirés d'un corpus du lien suivant, « https://sourceforge.net/projects/arabicstopwords »
- L'élimination de la vocalisation à l'aide d'une expression régulière, pour avoir la possibilité de rechercher un mot sans vocalisation.

Le nettoyage de donnée s'est fait pendant l'indexation des corpus, que nous allons détailler un peu plus tard.

```
def deNoise(text):
    noise = re.compile("""-    | # Tashdid
                           ً    | # Fatha
                           ٌ    | # Tanwin Fath
                           ُ    | # Damma
                           ٍ    | # Tanwin Damm
                           ِ    | # Kasra
                           ّ    | # Tanwin Kasr
                           ْ    | # Sukun
                           ـ    # Tatwil/Kashida
                       """, re.VERBOSE)
    text = re.sub(noise, '', text)
    return text
```

Figure 1 Fonction deNoise de la vocalisation

IV. Structuration de données (Balisage) :

Le balisage est une étape importante qui permet de structurer le corpus. Pour chaque genre nous avons adopté une nomenclature distincte :

```
<quran>
<surat id=" سورة هود">
<verse id="1">
الٓرٰ كِتَابٌ أُحْكِمَتْ آيَاتُهُ ثُمَّ فُصِّلَتْ مِنْ لَدُنْ حَكِيمٍ خَبِيرٍ
</verse>
.
..
</sura>
</quran>
```

Figure 2 Balisage du coran

```
<Hadith_charif>
<hadith id="1">
....
</hadith>
<hadith id="2">
....
</hadith>
</Hadith_charif>
```

Fiigure 3 Balisage du hadith

```
<author id="author1">
<poem-title id="title1">
<couple>
....    ......
</couple>
</poem-title>
<poem-title id="title2">
<couple>
....    ......
</couple>
</poem-title>
</author>
```

Figure 4 Balisage des poemes

```
<author id="author1">
<book-title id="title1">
<sentence id='1'>
....
</sentence>
<sentence id='2'>
....
</sentence>
</book-title>
<book-title id="title2">
<sentence>
....
</sentence>
</book-title>
</author>
```

Figure 5 Balisage des textes (nathr)

- Quran : Le balisage a été fait de sorte que le Quran se compose de plusieurs chapitres, chaque chapitre(soura) a un sous ensemble de versets (ayates).

- Hadith : La balise Hadith Charif englobe tous les hadiths existants.

- Poèmes : L'ensemble a été divisé par auteurs, chaque auteur ayant ses propres poèmes, chaque poème disposant de plusieurs couplets. Nous avons divisé chaque poème en plusieurs couplet pour extraire que le couplet lors de la recherche. Cette division s'est faite au moment de l'extraction du web.

- Textes : Le principe utilisé pour les poèmes se répète. Nous avons aussi divisé chaque paragraphe en plusieurs phrases (sentence). Les phrases ont été obtenu d'après la ponctuation du texte. Cela pour retourner à l'utilisateur que la phrase qui contient le mot. L'extrait aura donc certainement une sémantique.

V. Indexation

Nous avons opté de mettre nos connaissances acquises des 2 modules **Recherche d'information** et **base de données** pour effectuer une indexation des mots des corpus dans une base de données **SQL**, afin de rechercher un mot rapidement et avoir des exemples de phrases qui le contiennent.
Nous avons procédé de la manière suivante :

- Pour chaque corpus d'une période donnée, nous avons pris une balise et ses sous-balises.
- Pour chaque ensemble de balise, on recherche le texte contenu dans cette dernière, ensuite selon le type de texte (poème ou texte narratif etc. …), le texte est contenu forcement dans une balise nommée **sentence** pour texte narratif, ou **couple** pour un poème, donc on aura ces balises et leurs balises prédécesseurs.
- Ensuite on prend le texte d'une phrase ou d'un couplet, on le tokenize avec la fonction **tokenize** du module **Pyarabia**, on vérifie si c'est bel est bien un mot en arabe avec la fonction **is_arabicword** du même module **python**, on le nettoie avec la fonction **deNoise** et on le met

dans la base de données comme une clé primaire avec sa fréquence et les balises qui précisent sa position dans le corpus.
Exemple : le mot **أبصره** est situé grâce à l'ensemble des balises suivante
« text id=عصر الدول المتتابعة/author id=ابن الأبار/book_title id=إعتاب الكتاب/sentence id=124 »

Nous avons aussi décidé de mettre le corpus dans une base de données afin de faciliter l'accès à une phrase sachant que nous avons un ensemble de balises pour l'identifier et l'extraire du corpus.

Le schéma des deux tables indexe et le corpus dans la base de données et comme suit :

- Schéma table indexe : **(Mot, Mot_deVocalized, Fréquence, Ensemble de balise)**
- Schéma table corpus pour texte naratif : **(balise auteur, balise titre livre, balise sentence, phrae)**

Pour accéder à une phrase nous avons défini une fonction **GetSentence(mot,indexeDb,corpusDb)** qui étant donnée un mot, un indexe et un corpus dans une base de donnée, retourne les phrases qui contiennent le mot.

VI. Lemmatisation, PosTagging et Comparaison de phrases courtes :

Pour améliorer l'expérience de notre utilisateur sur notre plateforme, nous avons pensé à implémenter un algorithme simpliste pour le calcul de la similarité au niveau sémantique entre deux phrases, surtout qu'après avoir fait nos recherches nous n'avons pas trouvé d'outils le réalisant.

Nous n'ignorons parfaitement pas que ce dernier est un domaine de recherche assez vaste et qu'il s'agit de faire bien plus pour avoir des résultats consistants mais nous avons quand même décidé d'y contribuer avec nos modestes connaissances.

Après que l'utilisateur ait choisi un mot pour la recherche, il aura à partir du dictionnaire plusieurs définitions possibles ainsi que plusieurs extraits des corpus disponibles.

Pour lui simplifier la tâche et filtrer ces données, il choisira une définition et nous lui proposeront les extraits qui se rapprochent le plus de cette dernière.

Notre approche se base principalement sur deux paramètres :

- La similarité sémantique de l'extrait avec l'exemple donné dans la définition. Pour calculer

Celle-ci nous avons fait appel à la procédure de lemmatisation ainsi qu'aux synonymes du

Dictionnaire **el Maany** qui est assez riche.

Pour la lemmatisation nous avons utilisé **Farasa** qui est une boîte à outils de traitement de texte rapide et précise pour le texte arabe. Cet outil d'après des études statistiques est bien plus performant et exact que d'autres tels que **Madamira**, **Stanford**. [10] [11]

- La similarité de la structure de l'extrait avec celle de l'exemple donné dans la définition.

Pour calculer celle-ci nous ferons appel au **PosTagging.**

Nous avons utilisé l'outil proposé par **StanfordCoreNlp** en raison de sa rapidité vue que ce traitement doit être instantané pour l'utilisateur.

```
Fonction_Similatité_Semantique ( Liste syn_definition, String  quote) :
        sim=0 ;
        Pour mot dans quote :
            syno_quote.ajout( Synonymes( Lemmatize(mot)) ;
        Pour syn dans syno_quote :
            Si syn_definition.contient(syn)
            Alors sim+=1 .
            fsi
Retourner  (sim / longueur(quote))
```

```
Fonction Similarité_Syntaxique(String exemple_def , String  quote)
 sim=0 ;
 bigr_def= Bigrams(PosTagging(exemple_def))
 bigr_quote= Bigrams(PosTagging(quote))
 Pour (tag1,tag2) dans bigr_quote
        Si bigr_def.contient( (tag1,tag2))
        Alors sim+=1 ;
        Fsi ;
 Retourner ( sim/ longueur (quote))
```

```
Fonction Similarite(String def ,String quote) :
Retourner (Similarité_Syntaxique(def,quote)+ Similarité_Semantique(def,quote))
```

Les résultats obtenus avec cette approche ne sont pas encore satisfaisants mais encore il nous faut l'avis d'un expert pour pouvoir faire la conclusion.

VII. Application :

A. Etude du besoin de l'utilisateur :

Pour pouvoir comprendre les besoins des utilisateurs potentiels de notre future application et de s'en inspirer, nous avons mené une étude sur les réseaux sociaux.

Cela en publiant un formulaire sur différents groupes Facebook : d'enseignants de différents niveaux, de groupes d'étudiants universitaires spécialisés dans la langue arabe (de plusieurs pays) et des groupes de passionnés de la littérature arabe.

Malheureusement les personnes ayant répondu étaient juste satisfaites avec les fonctionnalités proposées et n'en exigeaient pas plus, ni en critiquaient. Lien : [12]

B. Outils utilisés :

Nous avons développé une plateforme web en utilisant Django et le langage de programmation python.

C. Fonctionnalités :

Dans ce qui suit nous exposerons les principales fonctionnalités de notre application

- **Extraction de définition, synonyme et antonyme :**

Afin de mener à bien cette tache nous avons utilisé deux bases de données différentes du dictionnaire **El Maany**, une première contenant les définitions et une seconde les synonymes et les antonymes des mots.

Pour extraire les données à partir des bases de données nous nous sommes servis du module SqlLite3 de python, l'ensemble de tuple sera ensuite traité et retourné selon les deux cas possibles :

- **Premier cas :** l'utilisateur entre un mot avec « techkil » le resultat est de suite retournée.
- **Deuxième cas :** l'utilisateur entre un mot sans « techkil », nous allons alors faire un second traitement qui consiste à retirer le techkil de tous les mots et de retourner les mots qui ressemble à celui recherché, l'utilisateur aura alors un résultat plus complet.

- **Visualisation des extraits des différents corpus :**

Une fois le mot défini, nous parcourons notre indexe pour visualiser les extraits des différents corpus. Vu que le nombre d'exemples est assez important nous en sélectionnons au maximum une trentaine de chaque corpus.

Pour diversifier l'ensemble et quand cela est possible, nous affichons un même pourcentage de chaque style littéraire (narratif, poème, hadith.

- **Visualisation de statistiques :**

L'utilisateur une fois les extraits affichés pourra apprécier les nuances de sens dans différents corpus. Nous avons pensé à enrichir cela par la fréquence d'utilisation du mot d'après l'ère lui indiquant ainsi la maitrise de ce mot par les personnes ayant vécues en cette époque.

- **Construction du dictionnaire historique :**

Une fois que l'utilisateur ait introduit l'objet de sa recherche et sélectionné le résultat qui lui convient, l'enregistrement se fait dans un fichier de format XML qui contient les informations suivantes :

- Le mot.
- La définition : extraite à partir de la base de données d'**El Maany** ou bien reformulée par l'utilisateur.
- L'exemple : sélectionné par l'utilisateur qui est obtenue à partir des corpus.
- L'ère à laquelle appartient l'exemple choisi par l'utilisateur.

- **Fonctionnalités facultatives :**

  - Traduction : Nous avons pensé au cas où l'utilisateur croiserait un mot étranger et voudrait connaitre son utilisation en arabe. La traduction se fait à partir de google traduction.
  - Virtualisation du clavier : On se retrouve souvent face au problème du clavier arabe, et les devinettes éternelles sur l'emplacement de telle ou telle lettre. Pour cela nous offrons à l'utilisateur un clavier virtuel arabe. Et parmi nos approches s'il désire écrire avec un clavier anglophone, toute lettre écrite sera convertie en lettre arabe comme le font les claviers en ligne tel que 'LEXILOGOS'.

VIII. Répartition des taches :

Chabane Imad

- Réalisation de la plateforme web et intégration des fonctionnalités du dictionnaire historique.

Bezgali Meriem

- Recherche et collecte des corpus des 3 périodes **Bni El-Abass, Dawla El-Moutatabia** et **moderne.**
- Extraction des 2 sites [6] [8] avec 2 scripts, ainsi que tentative d'extraction des fichiers en format PDF.
- Script d'ordonnancement automatiques des livres extraits.
- Balisage des corpus des texte narratifs et des journaux.
- Script d'indexation des différents corpus.
- Script de **GetSentence(mot,indexeDb,corpusDb)** pour retourner les phrases dans lesquelles se trouve un mot donné.
- Script de mise des corpus dans une base de données.
- Nettoyage des données.
- Participation à la rédaction du rapport.
- Contribution au mini-projet avec des idées.

Chennit Naila

- Recherche et collecte des corpus des 3 périodes **Bni El-Abass, Dawla El-Moutatabia** et **moderne.**
- Extraction des sites [5] [6] [8] avec des scripts
- Balisage des poèmes dans les corpus.
- Extraction en mode offline des définitions du dictionnaire El Maany
- Extraction en mode offline des synonymes et antonymes du dictionnaire El MaanyTadad
- Nettoyage des données.
- Script de construction du dictionnaire local (rempli par l'utilisateur)
- Script de génération des statistiques de la fréquence du mot.
- Participation à la rédaction du rapport.

Moussaoui Amina

- Recherche et collecte des corpus des 2 périodes : Islamique et Umayyad ainsi que le Coran
- Extraction des sites [4] [5] [6] [7] [8] avec des scripts
- Balisage du Coran, hadith.
- Nettoyage partiel de données
- Tentative d'extraction de texte arabe de fichier docx.
- Implémentation de la fonctionnalité de traduction online
- Implémentation de la recherche de définition du dictionnaire Maany online (non utilisé)
- Conception et Implémentation de l'algorithme de similarité de deux phrases.
- Script réalisant la lemmatisation et le posTagging avec l'outil Farasa
- Script réalisation le posTagging avec StanfordCoreNlp
- Participation à la rédaction du rapport.

Bibliographie et références :

[1] : https://fr.wikipedia.org/wiki/Traitement_automatique_du_langage_naturel

[2] : Description du mini-projet

[3] : https://ar.wikipedia.org/wiki/عصور_الأدب_العربي

[4] : https://equran.me/browse.html

[5] : http://www.poetsgate.com

[6] : http://shamela.ws/

[7] : https://www.mosoah.com/

[8] : http://www.alwaraq.net/

[9] : http://aracorpus.e3rab.com/

[10] : http://kareemdarwish.com/files/2017/10/28/farasa-fast-furious-arabic-segmenter/

[11] : http://www.jlcl.org/2017_Heft1/02-diacritics.pdf

[12] : https://docs.google.com/forms/d/e/1FAIpQLSfIN2WVJb0yB61s4dRVHmXQ8t2T14x70pmcVlJod0kk6MH3aA/viewform

Annexes :

Figure 2 Définition d'un mot, synonymes ,antonymes

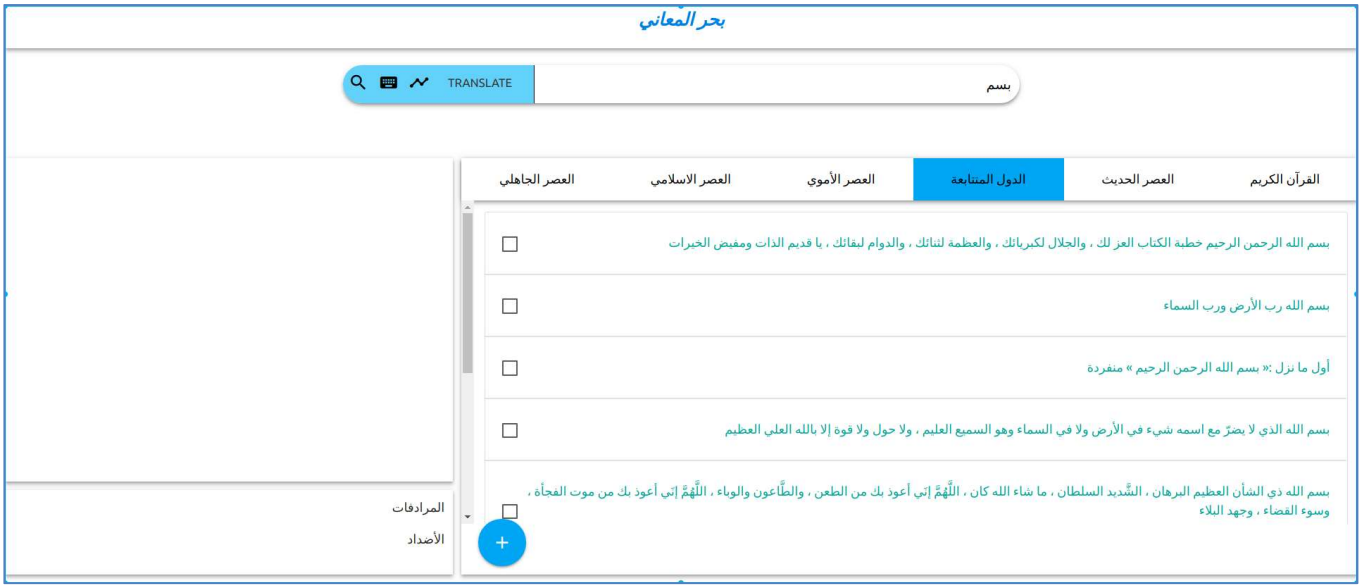

Figure 6 Définition d'un mot et extraction des exemples des ères consécutifs ((Al oosour el moutatabiaaa))

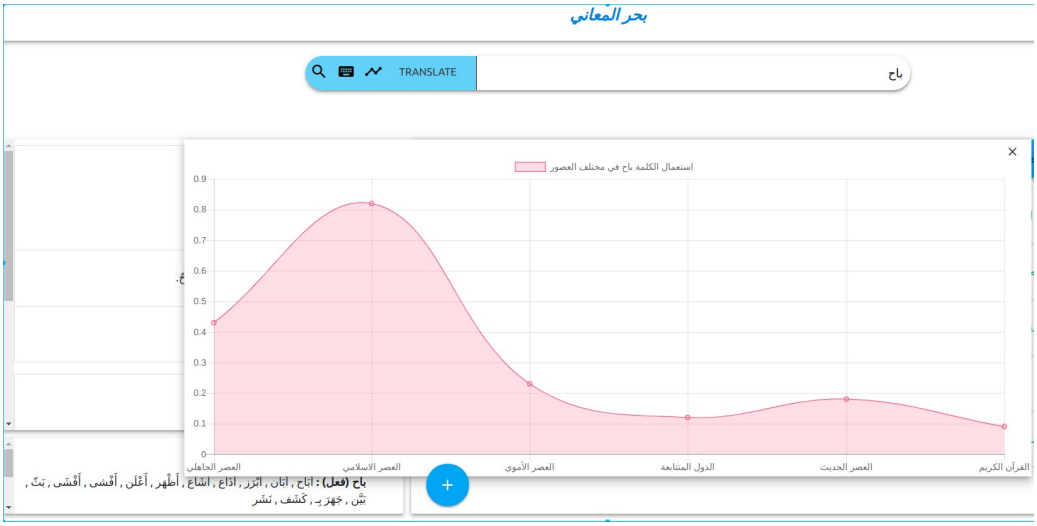

Figure 3 Visualisation de la fréquence d'apparition d'un mot d'après les ères