INF723 Dossier

12 novembre 2014

Résumé

Quick briefing about 華文. Introduce the context, what we plan to do and how to do it.

Ce document est une présentation du projet « 華文 ». Nous commencerons par décrire rapidement le contexte général de ce projet avant d'être plus spécifique et d'énumérer ce que nous aimerions pouvoir faire. Nous projetterons enfin quelques pistes de développement.

Table des matières

1 Enjeux et contexte

La langue chinois n'est pas comme le français : un caractère comme 貴 [1] peut signifier quelque chose en lui-même alors qu'une lettre comme *z* ne veut rien dire prise en dehors d'un mot. Le chinois est donc une langue très différente et donc très intéressante.

Mon but premier en commençant ce projet est de répondre à la question : « quel est l'ordre le plus facile pour apprendre les caractères chinois ? ». On peut en première approximation considérer qu'il suffit d'apprendre d'abord les composants d'un caractère avant d'apprendre le caractère lui-même. 貴 est par exemple composé de trois caractères : 中, 一 et 貝.

1.1 Type d'un sinogramme

Un sinogramme peut être classé selon son étymologie ou sa structure. Le type d'étymologie apporte un peu d'information à l'étudiant [2] et sa structure indique comment agencer les caractères qui le composent. Douze structures différentes existent et sont identifiées par les points de code du bloc Unicode `U+2FF0..U+2FFF` Ideographic Description Characters [3].

1.2 Etude de l'ensemble des sinogrammes

La plupart des sinogrammes sont dont composés d'éléments plus simples. Deux types de décompositions existent. Le premier considère qu'on peut décomposer un caractère comme un assemblage de traits de différents types. Cette façon de faire est pratique pour les calligraphes ou les typographes mais moins pour les étudiants.

Le second type de décomposition s'appuie sur l'étymologie et considère que certains caractères, les clefs, ne peuvent pas être décomposés. Dans cette optique, la clef 鼻, qui signifie *nez* ou *premier*, ne devrait pas se décomposer en 自, 田 et 丌 car cela n'a pas de sens. Nous utiliserons ce type décomposition dans la suite.

1. Signifie : honorable, précieux et se prononce *guì*.
2. Un sinogramme peut être un pictogramme (il représentent plus ou moins bien une forme, par exemple 木 représente un arbre), un idéogramme strict (本 désigne l'origine en montrant la racine d'un arbre) ou composé (明 réunit le soleil et la lune et désigne la lumière) ou encore un idéophonogramme (河 est composé de 氵, version déformé de l'eau 水 et de 可 qui indique la prononciation). Les lettrés chinois distinguent aussi les emprunts phonétiques et les dérivations.
3. Les structures disponibles sont : ⿰, ⿱, ⿲, ⿳, ⿴, ⿵, ⿶, ⿷, ⿸, ⿹, ⿺ et ⿻.

En décomposant les caractères, on obtient un graphe orienté acyclique que l'on peut appeler une forêt : il contient beaucoup d'arbres, chacun ayant pour racine un caractère composé et comme feuilles les clefs qui le composent.

Des questions plus pointues peuvent également être étudiées : peut-on énoncer un critère pour évaluer la pertinence d'un ensemble de clefs et déterminer un ensemble vérifiant ce critère de manière optimale ? Cette question demande de comprendre profondément la structure de la forêt. Elle peut être reformulé en termes ensemblistes : considérant une liste de clef comme une famille libre et génératrice (donc une base) du pseudo espace des caractères (l'ensemble des caractères n'est pas fini), peut-on trouver un critère d'évaluation de ces bases ? En imaginant que le critère soit la faible taille [4] d'une telle liste, peut-on trouver une base qui couvre un maximum de caractères avec le plus petit nombre de clefs [5] ?

1.3 Apprentissage du chinois mandarin

De même que le TOEIC ou le TOEFL permettent de juger le niveau d'anglais d'un étudiant, des examens similaires existent pour le chinois. Ces derniers proposent généralement plusieurs paliers de maîtrise de la langue, chacun étant associé à une base plus une moins large de caractères et chaque palier incluant les paliers précédents.

Etablir un ordre d'apprentissage des caractères est une question importante pour apprendre efficacement les caractères et progresser rapidement dans les tests de langues.

2 Cahier des charges

2.1 Finalité et fonctionnalités minimales

Cette description indique les fonctionnalités minimales qui devront être mises en œuvre à la fin du projet. Etant donné l'aspect incrémental du développement, il sera possible de rajouter plus de fonctionnalités ou d'améliorer l'existant.

Ce projet prendra en entrée un fichier texte contenant une liste de caractères chinois et leur décomposition. Il construira l'arbre de composition de ces carac-

4. L'analogie avec les espaces vectoriels trouve très rapidement ses limites car on ne peut pas considérer des bases qui n'aient pas toutes le même nombre d'éléments. Du coup, trouver une base revient à trouver la dimension de l'espace des caractères.

5. Les clefs ne sont pas forcément des sinogrammes associés à une sémantique mais cela faciliterait l'apprentissage. Sinon, on peut proposer comme base l'ensemble des traits possibles.

tères et produira en sortie deux fichiers : le premier sera une image au format `svg` de cet arbre et le second une séquence [6] de caractères pour répondre à la question « dans quel ordre apprendre le plus efficacement possible ces caractères » ?

La notion d'efficacité sera précisée en fin de projet : nous considérerons au minimum qu'une séquence est efficace si un caractère apparaît après tous ses composants.

2.2 Références

Je projette d'utiliser les éléments suivants :

- `maven` pour la gestion des dépendances ;
- `jooq` pour la liaison avec la base de données ;
- Un connecteur MARIADB pour la connexion à la base de données ;
- Une bibliothèque pour afficher et manipuler des graphes ;
- Une bibliothèque pour lire des fichiers `*.csv` et associer la valeur de chaque colonne au champ d'un objet java.

2.3 Rendu du projet

Ce projet a été initié il y a quelques temps par une phase de recherche bibliographique et un dépôt public github [7] a été créé pour faciliter la gestion de version. Il me semble que la manière la plus pratique de rendre mon travail est de prendre en compte la dernière révision de ce dépôt avant la date limite de rendu des projets. Ce dépôt permet aussi au correcteur de suivre l'évolution du projet.

Le projet rendu comprendra au minimum le code source du projet (en Java) muni de sa documentation (javadoc), le schéma de la base de donnée et des jeux de données de test. Un fichier d'aide détaillera comment compiler le projet à partir des sources et comment créer la base de données utilisée.

Le rendu du projet contiendra également un document d'analyse de l'approche orientée objet.

3 Analyse technique

Nous détaillons dans cette section quelques points qui nous semblent devoir être pris en compte pendant le développement de ce projet.

6. C'est-à-dire une liste ordonnée.
7. *cf.* `https://github.com/piotr2b/chinese-huawen`.

3.1 Traitement des données

Dans un premier temps, on stockera les caractères dans une simple structure de dictionnaire. Les arbres sont réalisés par les références des caractères. On suppose dans ce cas-là que tous les composants d'un caractère sont déjà connus quand celui-ci est analysé. La clef du dictionnaire sera la valeur de hachage de la chaîne de caractères contenant la description du sinogramme.

Dans un second temps, nous créerons une classe dictionnaire qui prend associe deux clefs `K1` et `K2` pour une valeur `V`. Il s'agira plus précisément de deux facettes d'une même clef puisque `K1` et `K2` seront liées et seront, dans le cadre de ce projet, la valeur de hachage de la chaîne de caractères contenant la description du sinogramme et le point de code du caractère représenté si disponible.

3.2 Schéma entité - association

Il faudra définir avec soin le schéma entité - association puis en déduire le schéma relationnel de la base.

3.3 Vérification de la qualité du projet

Des tests unitaires permettent d'écrire du code fiable.

3.4 Parallélisme

Le parallélisme apporte son lot de contrainte et de formalisme. Il faudra étudier s'il est souhaitable et possible d'écrire un algorithme parallèle. Dans cas-là, comment protéger la cohérence de la structure de données et éviter les doublons tout en profitant des hautes performances apportées par le parallélisme ?

3.5 Visualisation des données

Peupler une base de données ou faire pousser une forêt restent deux actions abstraites. Il serait souhaitable de trouver une bibliothèque pour afficher et manipuler des graphes pour produire une image de cette forêt.

Cependant, comment écrire un algorithme pour disposer harmonieusement la structure de données ?

3.6 Une modélisation plus complexe : rendre compte des variations

Les sinogrammes sont le fruit de huit mille ans d'histoire dont les aléas ont produits quelques particularités : ils peuvent être traditionnels ou simplifiés et peuvent avoir plusieurs variations [8] en fonction de la langue par laquelle il est utilisé. Les « sinogrammes » coréens diffèrent légèrement de leurs homologues chinois ou japonais. Réussir à unifier les variations d'un même caractère n'est vraiment pas évident mais permettrait de simplifier la sortie obtenue et généraliserait élégamment ce projet : il serait alors possible d'apprendre le chinois traditionnel ou le japonais (qui utilise des caractères simplifiés particuliers) sans mélanger les deux.

8. Le caractère 馬 qui signifie cheval est écrit 马 en chinois simplifié. Les caractères 戶, 户 et 戶 représentent le même sinogramme mais n'ont pas le même point de code Unicode (respectivement 6236, 6237 et 6238)