Weekly Reviews

Manon Rivoire

6^{th} May 2019

1 6^{th} May 2019 Review

1. I have studied the documentation of Bisect Module in Python and I have explained the working of this module in my LaTeX report.

2. I have implemented the display functions of the NFA and DFA in Python and made the display in console.

3. I have tried to install the appropriate packages and apply the GraphViz Library to Python implementation in order to visualize the automata under LaTeX. I have tested out the examples given by the GraphViz documentation in order to understand the working of this Library and to become more confident with this tool.

4. It remains some problems that I do not manage to solve even after many investigations. Among them, there are : a problem of reading of the states of the automaton in Python when we cross the automaton in order to display it and of conversion of these states labels to the LaTeX label format, another problems consists in the dimension and position of the automaton when we display it in LaTeX.

5. I have began to study the openFst Library.

6. I have to manage to better understand the working of the automata (NFA, DFA...).
   I have to use the openFst Library to perform automata and carry out matchings.
   I have to make the tutorials of MatchID.
   I also have to continue simultaneously the LaTeX report.

Objectives

- Report Commencer la biblio sur le record linkage (P3)
  Identifier un nom d'exemple
  Persévérer dans l'impression déjà faite : objectif comprendre latex et graphviz (comment afficher l'automate en graphviz et le paramétrer) (P2)

- MatchID :
  Tuto (P3)
  Comprendre l'algo actuel (P3)

- Finite State Automata Créer un graph avec openfst (P1)
  Faire l'export en .dot avec openfst (P1)
  Commencer la création de l'automate de Levenshtein avec openfst (P2)

2 14^{th} May 2019 Review

- Génération d'automates grâce à la librairie OpenFst
- Export en .dot et conversion en .tex pour visualisation sous LaTeX
- Création de l'automate de Levenshtein grâce à la librairie OpenFst en cours, problème d'affichage de l'automate au niveau de la forme sûrement dû à un problème soit de parcours de l'automate pour l'éditer soit d'indexation des noeuds
- Export en .dot et conversion en .tex pour visualisation sous LaTeX réussie
- Problème de taille de lors de l'affichage des automates sous LaTeX résolu avec le paramètre "scale = 0.5" directement dans le code LaTeX, l'attribut de taille dans le code du fichier .dot ne fonctionne pas.
- Conception du Matcher avec OpenFst en cours mais problème de compréhension quant au parcours de l'automate pour mettre à jour les labels avec la chaîne hypothèse passée en paramètres

Objectives

- Symbol Table pour afficher les labels des arcs du transducteur (P1)
  Possibilité de faire une fonction externe à la librairie convertissant le vocabulaire de l'automate en labels
- Ecrire la problématique sur les automates
- Fonction Fst To dot sous OpenFst
- Revoir les labels des arcs : ne pas se soucier de la chaine hyptothèse pour la construction de l'automate : construire les labels seulement sous la forme : n:etoile:1 c'est à dire en laissant les caractères généraux du vocabulaire (étoile, epsilon, et les caractères de la chaine de référence).
- fonction f.draw("f.dot") fonction permettant de générer le fichier .dot
- essayer de trouver d'autres librairies pour l'exécution d'automates intefacées en python (factorisation, exécution...)
- configurer visual code en version 2

3 21^{st} May 2019 Review

- Création de l'automate de Levenshtein sous forme d'un dictionnaire de profondeur 3
- Affichage de l'automate à partir de ce dictionnaire sous LaTeX grâce à Graphviz
- Création de l'automate de Levenshtein grâce aux fonctions de la librairie OpenFst ($Add_states()$, $Add_arc()$)
- Affichage de l'automate sous LaTeX grâce à la fonction draw de la librairie OpenFst
  Problème d'affichage au niveau des noeuds (disposition et label)
- Réflexion sur le matcher : arbre multinomial (parcours récursif de l'arbre ou itératif ?, utilisation de piles ?, obtention du sous-ensemble de mots reconnus par l'automate)
  Modélisation de l'arbre sous forme de listes de listes ?
- Parcours en profondeur (dfs = depth first search en récursif ? existe aussi en itératif)
- Pour la modélisation d'arbres : utilisation du module Arbre.py de python ?
- Difficulté à concevoir le matcher (faut-il prendre en compte à chaque niveau de profondeur de l'arbre toutes les possibilités ou privilégier les chemins de poids nul afin de minimiser le poids global de la chaîne reconnue ?)