Calcul de l'arbre recouvrant maximal (MST) de partie

Professeur référent : Michel Chilowicz <chilowi@u-pem.fr>

Nous expliquons dans ce document comment générer un arbre recouvrant maximal (*Maximum Spanning Tree : MST*) prenant en considération tous les mots proposés au cours d'une partie. La construction de cet arbre sera réalisée un peu différemment de celle de l'arbre recouvrant maximal du modèle. L'arbre recouvrant de partie ne comporte que les mots entrés par le ou les joueurs au cours de la partie.

Le MST de partie est généré itérativement à l'aide d'un programme Java que vous devrez réaliser.

Principe du jeu

Au démarrage de la partie, seuls deux mots sont présents sur le plateau : le mot de départ et le mot de fin. Un score est associé à ces deux mots. Pour l'instant l'arbre recouvrant maximal ne comporte que cette arête et le meilleur chemin connu ne fait qu'emprunter cette arête.

Le but du jeu est de proposer de nouveaux mots qui vont pouvoir enrichir l'arbre recouvrant maximal et possiblement générer de nouveaux meilleurs chemins se rapprochant le plus possible en terme de score du meilleur chemin théorique que l'on pourrait calculer par un solveur.

Processus itératif de construction

L'arbre recouvrant maximal pour les mots actuels que nous notons mst(w_1, w_2, ..., w_i) nous est communiqué par un fichier généré lors de l'itération précédente par notre programme. Nous disposons aussi d'un fichier de partie généré par le programme C de gestion de scores. Ce fichier de partie doit notamment comprendre les similarités calculées entre le nouveau mot w_{i+1} et tous les autres mots w_1, w_2, ..., w_i proposés lors des itérations de jeu précédentes.

L'objectif du programme que nous souhaitons réaliser ici est de calculer l'arbre recouvrant maximal constitué par l'ajout d'un nouveau mot w_{i+1} (mot proposé par le joueur). Nous pourrions faire le choix d'un algorithme recalculant de zéro l'arbre avec le nouvel ensemble de mots. Nous faisons plutôt le choix d'opérer itérativement (i.e. en prenant pour base l'arbre calculé lors de la précédente itération).

Algorithme de construction

L'algorithme de construction que nous décrivons aura pour tâche de calculer mst(w_1, w_2, ..., w_i, w_{i+1}), sachant que nous connaissons les éléments suivants :

- L'arbre recouvrant maximal mst(w_1, w_2, ..., w_i) de l'itération qui vient de s'achever. Cet arbre est exprimé dans un fichier généré lors de l'itération précédente par notre programme.
- Les similarités entre le nouveau mot w_{i+1} et tous les mots w_1, w_2, ..., w_i de l'arbre recouvrant actuel (ce qui représente i similarités à examiner). Ces informations sont présentes dans le fichier de partie généré par le programme C de gestion des scores.

Tout d'abord nous relions w_{i+1} à l'arbre recouvrant actuel en trouvant la plus forte arête contenant w_{i+1} : il faut donc connaître les similarités des i couples (w_{i+1}, w_1), (w_{i+1}, w_2), ..., (w_{i+1}, w_i).

Après l'insertion de la plus forte arête contenant w_{i+1}, peut on considérer que le nouvel arbre est un arbre recouvrant maximal des mots w_1, ..., w_i, w_{i+1} ? Nous n'en avons aucune garantie, en particulier si w_{i+1} dispose de très fortes similarités avec les nœuds déjà présent dans l'arbre. Il peut être alors nécessaire d'ajouter d'autres arêtes impliquant w_{i+1} dans l'arbre.

Nous procédons alors au tri de toutes les arêtes (w_{i+1}, w_1), (w_{i+1}, w_2), ..., (w_{i+1}, w_i) par similarité décroissante. Nous itérons sur ces arêtes dans cet ordre.

Lorsque nous nous intéressons à une arête (w_{i+1}, w_j), nous la raccordons dans l'arbre à w_j. L'arbre qui comportait i+1 nœuds avec i arêtes comprend alors désormais i+1 arêtes. Il ne s'agit donc plus d'un arbre puisqu'un cycle a été créé. Nous devons alors procéder à l'élimination de ce cycle.

Tout d'abord nous identifions le cycle ce qui peut être fait par un parcours de neuds à partir de w_{i+1}. Nous examinons toutes les arêtes constituant ce cycle avec leur similarité. Pour casser le cycle, on supprime une des arêtes du cyle : le choix portera sur l'arête de similarité minimale.

Si nous constatons que l'arête (w_{i+1}, w_j) possède une similarité plus petite que la similarité minimale entre deux nœuds de l'arbre, il est inutile de l'ajouter dans l'arbre : en effet lors de la suppression du cycle, ce sera ladite arête qui sera supprimée. Il n'est pas non plus nécessaire de continuer l'exécution de l'algorithme pour les arêtes plus petites qui suivent.

Exemple

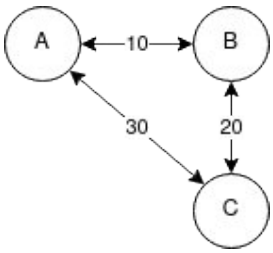
Démarrons un MST avec un sommet A.

1. Nous y ajoutons un sommet B avec une arête A —> B de similarité 10. Le MST contient alors les sommets A et B reliés par cette arête.

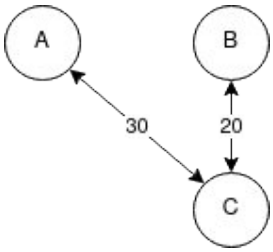
2. Nous ajoutons ensuite un sommet C avec les adjacences suivantes (par similarité decroissante) :

C	A	30
C	B	20

Nous relions C à A (30). Ensuite nous tentons de relier C à B. Nous obtenons un cycle :



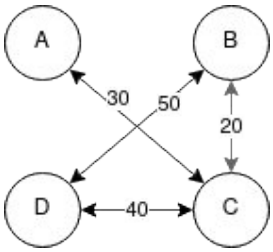
L'arête de plus faible similarité sur ce cycle est celle reliant A à B que nous supprimons :



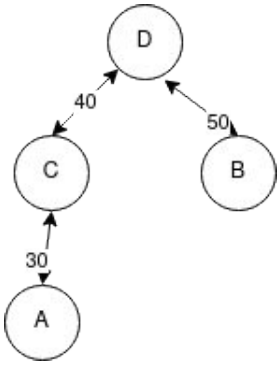
3. On continue avec l'ajout du sommet D dont les adjacences sont les suivantes :

D	B	50
D	C	40
D	A	5

On relie D à B. Puis on relie D à C ce qui induit un cycle que nous supprimons en enlevant l'arête C —> B :



L'arête D —> A peut être ignorée car sa similarité est inférieure à la plus faible arête du MST en construction. Nous obtenons finalement cet arbre :



Implantation

On attend une implantation de l'algorithme en Java. Comme on suppose que votre programme Java n'a pas accès directement aux données de similarité du modèle, il est nécessaire de les lui communiquer. On lui communiquera par des fichiers texte les données suivantes :

- L’arbre recouvrant maximal actuel provenant de l’itération précédente
- Le mot à ajouter avec similarités de ce mot avec tous les mots de l’arbre actuel

Le programme met à jour l’arbre en intégrant le mot indiqué ; il retourne sur la sortie standard l’arbre mis à jour avec le format adéquat dans un format similaire à l’arbre de l’itération précédente.

Vous devrez déterminer les formats des fichiers texte utilisés pour spécifier les MST ainsi que les similarités. Ces formats devront être explicités dans des documents livrés avec le projet.