#include <stdio.h>

#include <stdlib.h>

#include "tree.h"

#include "utils.h"

#include "data.h"

void change\_leaves(tree \*t, char \*leaf\_list)

{

list \*llist = get\_paths(leaf\_list);

char \*\*leaves = (char \*\*)list\_to\_array(llist);

int n = llist->size;

int i,j;

int found = 0;

for(i = 0; i < t->n; ++i){

t->leaf[i] = 0;

for(j = 0; j < n; ++j){

if (0==strcmp(t->name[i], leaves[j])){

t->leaf[i] = 1;

++found;

break;

}

}

}

fprintf(stderr, "Found %d leaves.\n", found);

}

float get\_hierarchy\_probability(float \*x, tree \*hier, int c)

{

float p = 1;

while(c >= 0){

p = p \* x[c];

c = hier->parent[c];

}

return p;

}

void hierarchy\_predictions(float \*predictions, int n, tree \*hier, int only\_leaves)

{

int j;

for(j = 0; j < n; ++j){

int parent = hier->parent[j];

if(parent >= 0){

predictions[j] \*= predictions[parent];

}

}

if(only\_leaves){

for(j = 0; j < n; ++j){

if(!hier->leaf[j]) predictions[j] = 0;

}

}

}

int hierarchy\_top\_prediction(float \*predictions, tree \*hier, float thresh, int stride)

{

float p = 1;

int group = 0;

int i;

while (1) {

float max = 0;

int max\_i = 0;

for (i = 0; i < hier->group\_size[group]; ++i) {

int index = i + hier->group\_offset[group];

float val = predictions[(i + hier->group\_offset[group])\*stride];

if (val > max) {

max\_i = index;

max = val;

}

}

if (p\*max > thresh) {

p = p\*max;

group = hier->child[max\_i];

if (hier->child[max\_i] < 0) return max\_i;

}

else if (group == 0) {

return max\_i;

}

else {

return hier->parent[hier->group\_offset[group]];

}

}

return 0;

}

tree \*read\_tree(char \*filename)

{

tree t = {0};

FILE \*fp = fopen(filename, "r");

char \*line;

int last\_parent = -1;

int group\_size = 0;

int groups = 0;

int n = 0;

while((line=fgetl(fp)) != 0){

char\* id = (char\*)xcalloc(256, sizeof(char));

int parent = -1;

sscanf(line, "%s %d", id, &parent);

t.parent = (int\*)xrealloc(t.parent, (n + 1) \* sizeof(int));

t.parent[n] = parent;

t.name = (char\*\*)xrealloc(t.name, (n + 1) \* sizeof(char\*));

t.name[n] = id;

if(parent != last\_parent){

++groups;

t.group\_offset = (int\*)xrealloc(t.group\_offset, groups \* sizeof(int));

t.group\_offset[groups - 1] = n - group\_size;

t.group\_size = (int\*)xrealloc(t.group\_size, groups \* sizeof(int));

t.group\_size[groups - 1] = group\_size;

group\_size = 0;

last\_parent = parent;

}

t.group = (int\*)xrealloc(t.group, (n + 1) \* sizeof(int));

t.group[n] = groups;

++n;

++group\_size;

}

++groups;

t.group\_offset = (int\*)xrealloc(t.group\_offset, groups \* sizeof(int));

t.group\_offset[groups - 1] = n - group\_size;

t.group\_size = (int\*)xrealloc(t.group\_size, groups \* sizeof(int));

t.group\_size[groups - 1] = group\_size;

t.n = n;

t.groups = groups;

t.leaf = (int\*)xcalloc(n, sizeof(int));

int i;

for(i = 0; i < n; ++i) t.leaf[i] = 1;

for(i = 0; i < n; ++i) if(t.parent[i] >= 0) t.leaf[t.parent[i]] = 0;

fclose(fp);

tree\* tree\_ptr = (tree\*)xcalloc(1, sizeof(tree));

\*tree\_ptr = t;

//error(0);

return tree\_ptr;

}