Project 4

(2015 Fall)

Course name: J1799d Instructor: Ming Li TA: Wenbo Liu

ID number	**15213796 / 15213782**	Name	**Luting Wang / Dajian Li**
Tel	**15622777988 / 13570300549**	Email	**364706267@qq.com / dajianl@andrew.cmu.edu**
Starting date	**2015.10.24**	Finished date	**2015.11.2**

1、 Project requirement

In this project, we are required to write a spell checker using lexical tree. We should first load a dictionary from dict_1.txt into the memory as a lexical tree, and then use this lexical tree with relative pruning of a beam width of 3 to correct typos in typos.txt, which contains segmented typos. For problem 2, we should adapt our lexical tree to automatically segment the text in unsegmented0.txt and unsegmented.txt. We are required to try relative beam widths of 5, 10, and 15 to find out which one works best.

2、 Design your program

In our program, we use an array to store the lexical tree. Because every single element in the array is a node of a tree, we have an array for each node to store indices of their children. Figure 1 demonstrates our lexical tree, given a dictionary “a, and, ate”. In order to load this dictionary, we first store ‘a’ in word “a” to the array at Index 1. When we are adding “and”, we find out that ‘a’ in “and” is an leaf node in the array, so we store ‘a’, ‘n’, ‘d’ at Index 2, 3, 4. A similar process happens when we are adding “ate”.

e		empty
t		6
d		empty
n		4
a		3 5
a		empty
*		1 2

Table 1

With this dictionary, we are now able to do the spell checking. We construct antrellis and use iterative method to calculate the costs in the trellis. To perform pruning, we have a copy of children of the nodes. Every time we prune out a word, we remove the index from the children array. There is a trick in implementing this method. For example, we would like to prune "ate", start from 'e', which is a leaf node, we find it's parent 't' which has only one child, so we will not perform pruning at 't'. Instead, we find 'a', the parent of 't', and find out that it has more than one children, so we remove 5 from it's children. In this way we can prune the word completely

For Problem two, the adaption is every time we meet a leaf node in trellis, we calculate its cost back to "*", and add this "lookback" cost and least node cost together to the next character's * state. This state is served as a transfer station and identification for inserting a space when we trace back from the last node.

r	1	2	1	3
o	1	0	1	2
d	3	2	1	0
r	2	1	0	1
o	1	0	1	2
w	0	1	2	3
*	0	2	1	2
	w	o	r	d

Figure 1

Figure 1 shows an example of such searching trellis. Supposed we have a dictionary including: "word", "wo" and "r", and we a looking for "word". From the last line, we find the minimum leaf node cost "0", and trace back. If we meet a node whose parent is "*", we insert a space. And finally we can get the word corrected and segmented.

3、 Program implementation and testing.

Data structure:

vector<Dict> lexicalTree stores characters in the dictionary and their children.
vector<Dict> temp_lexicalTree create a template of lexical tree to perform pruning.
vector<wordLookUp> dictionary stores each word and its endpoint index to print out.
vector<vector<Trellis> > trellis searching trellis, contains the information each node's cost, and pointers indicating where it is from.

Functions:

void readDictionary() load the dictionary by read the word and construct lexical tree.
bool search_trellis_pruning (int index, string s, int i, int mincost, int &mincost_next, bool &dirty) iterative functions that calculate trellis in Part 1. The returned Boolean value is a helper flag for pruning.
levenshtein_lexicalTree(string s, int &wordIndex) will call search_trellis_pruning and find out the best word in Part 1.
bool search_trellis_lookback_pruning(int index, char c, int n, int mincost, int &mincost_next, int &mincost_leaf, bool &dirty, int &mincost_leaf_index) iterative functions that calculate trellis

in Part 2. The returned Boolean value is a helper flag for pruning.
void levenshtein_lexicalTree_lookback(char c, int &n, int &wordIndex, bool &unsegment, int &mincost, int &mincost_next) will call search_trellis_pruning_lookback and print word and space in Part 2.
void levenshtein_computeAccuracy(vector<string> s1, vector<string> s2, int flag, int &in, int &de, int &sub) will compare the segmentation and corrected spelling and calculate accuracy.

4、 Experimental results and discussion

Figure 2 is the result of problem 1. As you can see, the lexical tree works, and the result is reasonable. We use a relative pruning with a beam width of 3. Since we are not given the correct template, so we didn't calculate its accuracy. For details, please refer to "spellchecked_typos.txt".

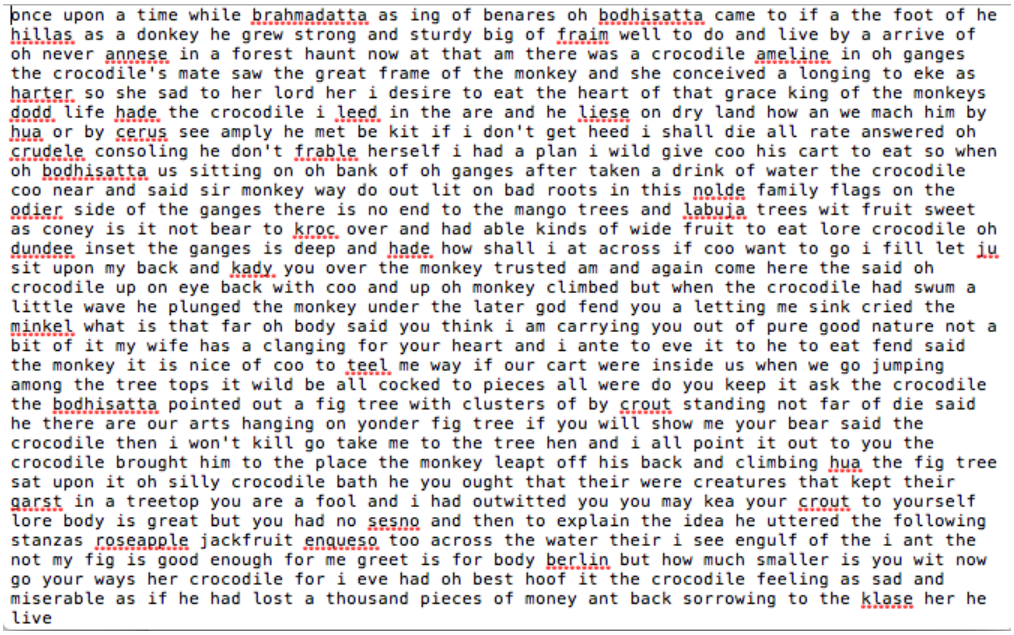
once upon a time while brahmadatta as ing of benares oh bodhisatta came to if a the foot of he
hillas as a donkey he grew strong and sturdy big of fraim well to do and live by a arrive of
oh never annese in a forest haunt now at that am there was a crocodile ameline in oh ganges
the crocodile's mate saw the great frame of the monkey and she conceived a longing to eke as
harter so she sad to her lord her i desire to eat the heart of that grace king of the monkeys
dodd life hade the crocodile i leed in the are and he liese on dry land how an we mach him by
hua or by cerus see amply he met be kit if i don't get heed i shall die all rate answered oh
crudele consoling he don't frable herself i had a plan i wild give coo his cart to eat so when
oh bodhisatta us sitting on oh bank of oh ganges after taken a drink of water the crocodile
coo near and said sir monkey way do out lit on bad roots in this nolde family flags on the
odier side of the ganges there is no end to the mango trees and labuja trees wit fruit sweet
as coney is it not bear to kroc over and had able kinds of wide fruit to eat lore crocodile oh
dundee inset the ganges is deep and hade how shall i at across if coo want to go i fill let ju
sit upon my back and kady you over the monkey trusted am and again come here the said oh
crocodile up on eye back with coo and up oh monkey climbed but when the crocodile had swum a
little wave he plunged the monkey under the later god fend you a letting me sink cried the
minkel what is that far oh body said you think i am carrying you out of pure good nature not a
bit of it my wife has a clanging for your heart and i ante to eve it to he to eat fend said
the monkey it is nice of coo to teel me way if our cart were inside us when we go jumping
among the tree tops it wild be all cocked to pieces all were do you keep it ask the crocodile
the bodhisatta pointed out a fig tree with clusters of by crout standing not far of die said
he there are our arts hanging on yonder fig tree if you will show me your bear said the
crocodile then i won't kill go take me to the tree hen and i all point it out to you the
crocodile brought him to the place the monkey leapt off his back and climbing hua the fig tree
sat upon it oh silly crocodile bath he you ought that their were creatures that kept their
garst in a treetop you are a fool and i had outwitted you you may kea your crout to yourself
lore body is great but you had no sesno and then to explain the idea he uttered the following
stanzas roseapple jackfruit enqueso too across the water their i see engulf of the i ant the
not my fig is good enough for me greet is for body berlin but how much smaller is you wit now
go your ways her crocodile for i eve had oh best hoof it the crocodile feeling as sad and
miserable as if he had lost a thousand pieces of money ant back sorrowing to the klase her he
live

Figure 2

For Problem 2, the results are shown in Figure 3 and 4. For the text with version with correct spelling, the accuracy is high, about 97%. First, we set the "lookback" penalty as 0, and tried different beam widths of 5, 10 and 15, the accuracy is shown below.

Relative pruning bandwidth	Total number of word	Insertion	Deletion	Substitution	Total errors	Accuracy
5	164	3	0	6	9	94.51%
10	162	1	0	4	5	96.91%
15	162	1	0	4	5	96.91%

Table 2

As you can see, although we apply pruning with a band width of 3, the accuracy is still acceptable. Next we try different "lookback" penalty (1, 2, 3), the accuracy is decreased. So we pick up 0 as the "lookback" penalty. The segmented version is shown in Figure 3. For details, please refer to "spellchecked_unsegmented0.txt".

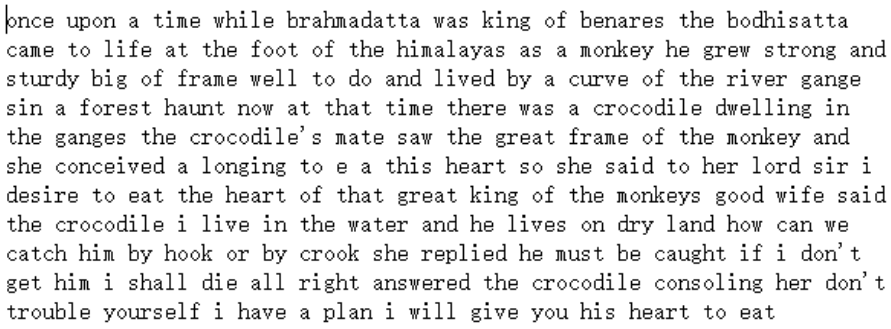

Figure 3

For the text with version with wrong spelling, the accuracy is much lower. For different beam width, the accuracy is shown below.

Relative pruning bandwidth	Total number of word	Insertion	Deletion	Substitution	Total errors	Accuracy
5	207	44	1	78	123	40.58%
10	207	44	1	78	123	40.58%
15	162	44	1	78	123	40.58%

Table 3

Next we try different "lookback" penalty (1, 2, 3), the accuracy is decreased. So we pick up 0 as the "lookback" penalty. The segmented version is shown in Figure 4. For details, please refer to "spellchecked_unsegmented.txt".

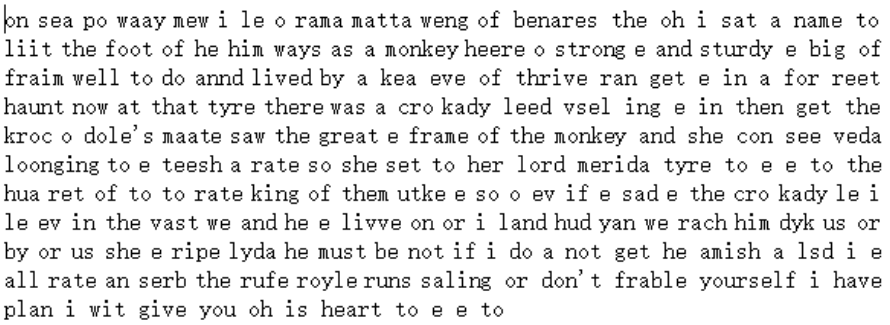

Figure 4

What is also worth noticing is that if we try to pick up the node with last minimum value instead of the one with first minimum value, the accuracy would be improved about 5%. For better segmentation, I would suggest to include some word transition cost to the edge cost, and delete some disturbing and meaningless word in the dictionary.