Tag Language Model Report

Bin Yuan
December 8, 2014

1 INTRODUCTION

Building a language model usually requires a large amount of training data, which is burdensome to obtain. And it is impractical to construct a language model that covers an entire spoken language, including specialized and technical fields. When there are some words and we don't have the corresponding corpus, it's impossible to recognize them for the word based language model. We combine the class based language model and word based language model using WFST(weighted finite state transducer) composition operation. The class we used in our method is a specific named entity(such as address name, person name etc.) and we call the composite language model as tag language model. We explore the relation between the composition weight and other hyper-parameters. We test our method on a domain-specific test set and our approach can dramatically improve the recognition rate of the low-frequency or unseen words in the training corpus.

2 METHOD

We test our approach in the domain of recognizing address names. The main procedure is as follows:

1. extract the low-frequency address names from training corpus and build a jsgf(JSpeech Grammar Format) file[1].
2. add tag to training corpus using the low-frequency address names and create a new corpus.

[1] http://www.w3.org/TR/jsgf/

3. compile the new corpus and the jsgf file respectively to G.fst and grammar.fst.

4. compose G.fst and grammar.fst to generate merged_G.fst.

5. construct HCLG using merged_G.fst.

We use NER(named entity recognition) to recognize the address names in the training corpus and classify the low-frequency address names to a class named "ADDRESS". Then we add some unseen address names to the class. We define the "ADDRESS" class by a rule in a jsgf file. Later we use the low-frequency address names to tag the training corpus, that is substituting the address name in trainning corpus with a "<address>" tag. A n-gram language model is generated for class tokens using the tagged training corpus. Compiling the n-gram language model and the jsgf file respectively to G.fst and grammar.fst. Fst compilation done, we compose the two fsts to generate a merged_G.fst parameterized by the composition weight. After the composition procedure, the merged_G.fst serves as a component of constructing HCLG.

2.1 Address Name Recognition

We use the ltp sdk[2] to recognize the address name in the training corpus. Extracting all the address names in the training corpus using the ltp tool. After Counting and sorting the address names by frequency, we set a frequency threshold and select the address names below this threshold. We classify the selected address names to the "ADDRESS" class and add some unseen address names to this class. Using the selected low-frequency address names and unseen address names to construct a jsgf file. The form of the grammar file is as follow:

```
#JSGF V1.0;

grammar address_grammar;

public <address> = low_freq_address_1 | low_freq_address_2 | low_freq_address_3 | un-
    seen_address_1 | unseen_address_2 | unseen_address_3;
```

In this file, line 1 is the JSGF version, and line 2 specifies the grammar name of the grammar file. Line 3 is a rule named "<address>" defines the "ADDRESS" class. The preceding address names of this rule is the low-frequency part, while the posterior part are the unseen address names that we add according to our need.

2.2 Tag the Training Corpus

The training corpus used relates with the area of telecom customer service, and it has a total size of 64 MB and consists of about 1.4 million setences. We use the extracted low-frequency address names to tag the training corpus. Traversing the training corpus and judging if a sentence contains any low-frequency address name. If does, we would create a new sentence with hit address name replaced by a "<address>" tag. Then we add the new sentences to the

[2]https://github.com/HIT-SCIR/ltp

original corpus to generate a new corpus called tagged corpus. For example, if "Beijing" is a low-frequency address name, and a sentence in corpus is "I live in Beijing", then a new sentence "I live in <address>" is added to new training corpus. Following is an example:
low-frequency address name list:

Beijing

original training corpus:

I live in Beijing

tagged corpus:

I live in Beijing

I live in <address>

2.3 FST COMPILATION

We use the tagged corpus to train a 5gram language model using SRILM toolkit[3]. The 5gram language model is then compiled to G.fst using OpenFst toolkit[4].
We use sphinx_jsgfsg toolkit and OpenFst toolkit to convert the jsgf grammar file to grammar.fst. Usage is as follow:

sphinx_jsgf2fsg -fsm address.jsgf address_grammar.address fsg.txt sym.txt

fstcompile –isymbols=sym.txt –acceptor=true < fsg.txt >grammar.fst

In the first step, jsgf format grammar file "address.jsgf" is converted to "fsg.txt", which is in text format. In the meantime, a symbol table "sym.txt" of the jsgf file is generated. In the second step, the text format "fsg.txt" is compiled to binary format "grammar.fst".

2.4 FST COMPOSITION

We compose G.fst and grammar.fst to generate a merged_G.fst. The fst composition procedure is parameterized by the composition weight. When the composition weight is small, it's more likely to recognize the address names in jsgf grammar file. Usage is as follow:

./G_merge.sh merge.cfg

The G.fst and grammar.fst is specified in the configuration file "merge.cfg", as well as the composition weight. The form of configuration file is as follow:

#tag label added to training corpus

target_label='<address>'

[3]http://www.speech.sri.com/projects/srilm/
[4]http://www.speech.sri.com/projects/srilm/

#composition weight of G.fst and grammar.fst

merge_weight=

#path of G.fst

source_G_big=

#symbol file path of G.fst

source_sym_big=

#path of grammar.fst

source_G_small=

#symbol file path of grammar.fst

source_sym_small=

#path of original lexicon

source_lex=

#the composition of G.fst and grammar.fst

target_G=

#generated new symbol file

target_sym=

#generated new lexicon

target_lex=

The G_merge.sh read the configuration file and generate three new files, merged_G.fst, a new symbol file and a new lexicon. There are some special handlings we should do. First, we should delete the item "<address> SIL" item in the new lexicon; second, we should delete the item "#0" at the end of the new symbol file.

2.5 HCLG CONSTRUCTION

We use Kaldi toolkit[5] to construct HCLG, and the input G is the merged_G.fst. The acoustic model we used here is mdl_1400. The construction command is as follow:

sh kickoff.sh cfg hclg_output_dir acoustic_model_dir

cfg is the configuration file of constructing HCLG, the form of cfg is as follow:

[5]http://kaldi.sourceforge.net/

#phone list

ph=

#arpa format or fst format language model path

lm=

#symbol file

sym=

lexicon file

lex=

In the baseline experiment, we use arpa format language model and don't need to specify the symbol file. But in the tag language model experiments, we use fst format language model(lm=merged_G.fst) and assign sym with the new symbol file path and lex with the new lexicon file.

3 EXPERIMENT

3.1 TEST SET

In order to test the perfomance of our method, we construct a test set. It contains 12 person's record of total 120 sentences. Each sentence contains a address name. The preceding 30 sentences have high-frequency address name in each sentence, and the middle 40 sentence have low-frequency address name which is listed in the jsgf grammar file in each sentence, while the posterior 50 sentence have address name that is unseen in corpus but added to the jsgf grammar file. We add 10 unseen address names to the jsgf file, and each unseen address name have 5 test sentences. We want to see whether our method can help us recognize the low-frequency address names and the unseen address names on condition that don't influence the recognition of other words.

3.2 BASELINE

We set up a baseline experiment to compare with our method. We use the seed corpus to train a arpa format 5gram language model. Using this language model to construct HCLG and testing on the test set. The result is in table 3.1. In the baseline experiment, the word error is

Table 3.1: Result of baseline

WER	Ins	Del	Sub	30_ER	40_ER	50_ER
20.66 [848 / 4104]	189	354	305	6	16	32

20.66%, 6 errors in recognizing the high-frequency address names, 16 errors in recognizing the

low-frequency address names and 32 error in recognizing the unseen address names. Note that there are 18 correct case when recognizing the unseen address name. The reason for this result is that we add 2 long address names each contains several words to grammar file. Although they don't appear in training corpus, but each single word compose the long address name appears in the training corpus.

3.3 TAG BASED LANGUAGE MODEL

We conduct 3 experiments to show our method works and to find the relationship between the optimal composition weight and the address names number grammar file. In experiment 1, we extract 490 low-frequency address names from seed corpus, using these 490 address names to tag the seed corpus, total 1369 sentences added to training corpus, combining the 490 low-frequency and 10 unseen address names in a grammar file. Using our method to build several tag based HCLGs with different composition weight, and the result is in table 3.2. From the above table we can see that our method can largely improve the recognition

Table 3.2: Result of experiment 1

composition weight	WER	Ins	Del	Sub	30_ER	40_ER	50_ER
0.1	16.72 [686 / 4104]	264	230	192	8	3	5
0.3	15.42 [633 / 4104]	182	288	163	7	3	5
0.5	15.40 [632 / 4104]	172	300	160	6	4	7
0.6	15.28 [627 / 4104]	166	302	159	6	7	6
0.7	15.28 [627 / 4104]	147	313	167	6	8	7
0.8	15.38 [631 / 4104]	154	296	181	6	7	9
1.0	15.98 [656 / 4104]	169	291	196	6	8	12
2.0	19.08 [783 / 4104]	172	377	274	6	14	23

rate of low-frequency and unseen address names, especially when the composition weight is small. But when the composition weight is smaller than a threshold, the WER is high. The explanation for this phenomenon is that when the composition weight is small, it's more likely for the decoder to recognize other words as address names in the grammar file. When the composition weight is high, the decoder is less likely to recognize the address names in the grammar file. One extreme is that when the composition weight is very big, the result of our method is same with baseline. There is a trade-off between the precision of low-frequency or unseen address names and other words. In experiment 1, we find the optimal composition weight is 0.6.

3.4 OPTIMAL COMPOSITION WEIGHT EXPLORATION

In order to find the relation between the optimal composition weight and the size of the grammar file, we conduct experiment 2 and experiment 3.
In experiment 2, the grammar file is same with experiment 1. But we use 100 low-frequency

address names to tag the training corpus, and add 165 sentences to corpus. The only difference between experiment 2 and experiment 1 is the number of sentences containing "address" tag in corpus. The result of experiment 2 is in table 3.3. In experiment 3, we use 490 low-frequency

Table 3.3: Result of experiment 2

composition weight	WER	Ins	Del	Sub	30_ER	40_ER	50_ER
0.01	17.57 [721 / 4104]	262	279	180	10	4	7
0.03	17.25 [708 / 4104]	257	277	174	8	4	7
0.05	16.84 [691 / 4104]	251	270	170	7	4	7
0.08	16.59 [681 / 4104]	269	234	178	6	5	8
0.1	16.50 [677 / 4104]	249	267	167	6	5	9
0.15	16.76 [688 / 4104]	244	274	170	6	6	11

address names to tag the training corpus and add 1369 sentences to corpus. But we use a different grammar file. This grammar file contains 1280 address names, 500 address names are from the grammar file in experiment 1 and remaining 780 address names are unseen in corpus. The result of experiment 3 is in table 3.4. Comparing experiment 2 and experiment 1,

Table 3.4: Result of experiment 3

composition weight	WER	Ins	Del	Sub	30_ER	40_ER	50_ER
0.1	17.42 [715 / 4104]	251	257	207	8	3	8
0.3	15.74 [646 / 4104]	196	268	182	7	3	6
0.5	15.20 [624 / 4104]	162	297	165	6	4	7
0.6	15.11 [620 / 4104]	162	298	160	6	6	6
0.7	15.52 [637 / 4104]	158	301	178	6	8	7
0.8	15.30 [628 / 4104]	150	299	179	6	8	8
1.0	15.69 [644 / 4104]	130	355	159	6	10	8
1.5	17.37 [713 / 4104]	158	327	288	6	12	15

we can find that when we add more sentences containing tag to training corpus, the optimal composition weight tends to bigger. Comparing experiment 3 with expriment 1, we can find that the number of address names in the grammar file don't influence the optimal composition weight obviously.

4 CONCLUSION

We combine the class based language model and word based language model. We classify the low-frequency address names extracted from the training corpus to a class ADDRESS and each single word is a class, thus there are V + 1 classes(V is the vocabulary size). The class based and word based language model is combined by fst composition guiding by composition weight. The result of our experiment shows that using a appropriate composition weight, the recognition rate of low-frequency and unseen words can be highly improved without

obviously reducing the recognition rate of other high-frequency words. In addition, we explore the relation between the composition weight and number of sentences added to corpus, as well as the number of address names in the grammar file.