# LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION IN TELEHEALTH

A Thesis presented to the Faculty of the Graduate School

University of Missouri-Columbia

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

by

XIAOJIA ZHANG

Dr. Yunxin Zhao, Project Supervisor

DECEMBER 2005

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled.

LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION IN TELEHEALTH

Presented by Xiaojia Zhang

A candidate for the degree of Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

[illegible] 11/16/05

[illegible]

[illegible] 11-22-05

## ACKNOWLEDGEMENTS

It is a long and meaningful journey for me to get the research work done for my master's degree in computer science. Finally, mission accomplished. I want to express my thanks to my dear professor and folks who helped me in this project.

I gratefully express my sincere appreciation to Dr. Yunxin Zhao for her guidance, advice, support, patience and endless kindness during the course of research work and thesis writing. Without her constructive advice and insightful comments, there is no way for me to go through this master's program and accomplish the job.

I also would like to thank Dr. Shi and Dr. Schopp for their support and service as my committee. It was with their advice and comments that I finished this thesis at last.

I want to give special thanks to my parents for their continuing encouragement and support. Their encouragement and caring were invaluable on my way pursuing my education and career. And I sincerely thank my husband Dr. Hongbing Jiang for his help during my graduate studies.

At last, I want to thank my fellow research group mates, from whom I got lots of advice and help.

LANGUAGE MODELING FOR AUTOMATIC SPEECH RECOGNITION IN TELEHEALTH

Xiaojia Zhang

Dr. Yunxin Zhao, Thesis Supervisor

## ABSTRACT

Standard statistic n-gram language models play a critical and indispensable role in automatic speech recognition (ASR) applications. Though helpful to ASR, it suffers from a practical problem when lacking sufficient in-domain training data that come from same or similar sources as the task text. In order to improve language model performance, various datasets need to be used to supplement the in-domain training data. This thesis investigates effective approaches to language modeling for telehealth which consists of doctor-patient conversation speech in medical specialty domain. Efforts were made to collect and analyze various datasets for training as well as to find a method for modeling target language. By effectively defining word classes, and by combining class and word trigram language models trained separately from in-domain and out-of-domain datasets, large improvements were achieved in perplexity reduction over a baseline word trigram language model that simply interpolates word trigram models trained from different data sources.

TABLE OF CONTENTS

LIST OF TABLES

# Chapter 1

# Introduction

## 1.1 Language Modeling in General

Knowledge about languages is important to many fields, including spoken language recognition and natural language understanding. The knowledge involves 1) lexical knowledge that is about the word definition and pronunciation, 2) syntactical knowledge that describes the structure of a language, and 3) semantic knowledge that shows meaning of words and phrases. Language modeling, whose task is to estimate the possibility of occurrence of a sequence of words based on pre-acquired knowledge, is a critical part of many applications, including ASR (automatic speech recognition), OCR (optical character recognition) and SMT (statistical machine translation). There are two approaches for modeling languages: formal language model approach and stochastic or statistical language model (SLM) approach. The former is also known as a rule-based approach, which explores a set of language grammars as rules, and parses sentences and analyzes sentence structural compliance with the rules. This approach is relatively difficult to acquire automatically, and needs manual working from linguists, since rules need to be generalized and explained. In addition, in real conversational applications, spoken language can be very ungrammatical and not follow any conventional rules, and it is difficult to measure the appropriateness of a sentence by matching it to manually established grammatical rules. The other approach, stochastic language modeling (SLM), is different from rule-based modeling approach in that it is a way to measure word sequence probabilities based on statistical information acquired from language itself, without involving manually-defined rules. It has been continuously improved during the past decade and is currently used widely. The goal of SLM approach is generally to measure the probability of sentences and give higher possibility value to a sentence that is more likely to appear in natural language. The probability value is calculated using the probabilistic information carried by statistic parameters that were automatically learned from a reasonably large and topic related training data set. These statistic parameters form a language model, which can be formulated as a probability distribution $P(W)$ over word string $W$. It is a reflection of how possible the word string can appear as a sentence. Unlike rule-based models, grammaticality is not a quite strong constraint to the stochastic language models. Even if a word string is not grammatical, it is still possible for such a model to assign the word string a high possibility. Here we use a very commonly used expression as example. In casual speech, people often say “I am doing good,” which will be recognized as an incorrect expression in formal English, because it uses an adjective “good” as a adverb to modify a verb “doing”. Although it is not grammatically correct, this word sequence appears in daily spoken language quite frequently. Using Switchboard, a conversational styled text dataset we used for language model training, as target corpus, we see this kind of incorrectly used phrase “doing good” 14 times in the whole dataset, while the total number of the expressions that should be used as “doing well” was only 36. This is to say that out of 36 times of the expression that should be said as “doing well”, only 22 instances of this expression were used grammatically correctly. Since this language modeling problem we present in this thesis is for an ASR project on telehealth, whose speech style is conversational, we chose and used SLM approach, hoping its power of automatic statistic knowledge learning will help with a satisfying recognition results.

## 1.2 Language Model in ASR

In applications using language models, generally the tasks can be described as finding the word string that most likely matches a sequence of observations. The task is defined as:

$$\widehat{W} = \arg\max_{w} P(W \mid X) = \arg\max_{w} \frac{P(W)P(X \mid W)}{P(X)} = \arg\max_{w} P(W)P(X \mid W) \qquad (1)$$

In OCR, $X$ in this formula is a set of handwritten characters; to SMT, $X$ is the target language needing translation; while in ASR, $X$ is an acoustic observation. Focusing on ASR problem that our language model is used for, we are to find the word string that mostly possibly is the transcript of certain acoustic observation. In the formula for this specific task, $W$ is any possible word sequence, $X$ is the acoustic observation, $\widehat{W}$ is estimated result. ASR problem can be finally decomposed to two parts, language modeling part $P(W)$ and acoustic part $P(X \mid W)$ corresponding to formula (1).

In this thesis, I will focus on language modeling part $P(W)$.

## 1.3 Project Motivation

Statistic  language modeling (SLM), whose task is to estimate the possibility of occurrence of sequences of words based on knowledge acquired from training text corpora, plays an important role in automatic speech recognition application. N-gram language modeling, which is the most commonly used SLM technique, predicts the probability of occurrence of a sentence by using a Markov process. During the past decade, word n-gram language models have proved to be extremely powerful when an adequate amount of relevant training data is available for parameter estimation. However, low-frequency or unseen events cannot be reliably estimated in many applications, such as specific topic tasks and conversational speech style tasks, where in-domain text materials are usually limited in amount and costly to produce. Therefore, class-based n-gram approaches have been developed by various research groups to deal with the sparse data problem with word classes defined using various classification techniques [2, 4, 13, 15]. Although class models suffer from problems such as not being portable, they have been accepted as good models for being able to well estimate unseen n-grams and especially helpful in some domain-dependent tasks and in some tasks involving small size dataset with a compact domain. In addition, techniques involving using large amount of out-of-domain data to improve in-domain language models have been proposed and some methods have been shown effective in certain domain-dependent problems [4, 14].

Methods to define word classes are an active area of current research. Among these methods of word clustering proposed for class-based language models, some of them are useful, such as automatic bottom-up greedy clustering [2], hierarchical word clustering [15] and part-of-speech (POS) tagging method [9]. In this thesis, we will discuss some of these methods as well as a way we used to define word classes for subsets of words with clear medical semantics, and some other word categories including names of people, numbers, etc., for the telehealth language modeling task. A similar approach was proposed in [11] for ATIS task. Experiments were also done on clustering methods, and compared the performance of language models based on semantic classes and the classes derived from automatic clustering.

The language modeling techniques that will be discussed in this thesis is for automatic recognition of conversation speech in telehealth system. Telehealth is a videoconference technique used in health care. With the system, patients, even staying at home, can get a high quality medical care taking advantages of audio/video technologies. This saves the time and cost for elderly people or people living in rural areas. However, because of possible audio/video delay and audio quality problem, people with hearing barriers could hardly take advantages of the current version of telehealth system. Adding a captioning function became necessary in order to perfect the telehealth system. The goal of our automatic speech recognition project in telehealth is to capture the doctor side conversation for recognition; hence the language model is focused on doctor's speech. The captions of doctor's speech will be provided to patients to help those with hearing disabilities achieve better understanding of doctors' questions and explanations. The described task is a specific medical domain problem involving spontaneous dialog style speech, and there are little existing in-domain text corpora that we can directly use for language modeling. At the current stage, the telehealth conversation transcriptions we have collected are still insufficient for reliable language model training. Therefore, we need to use large amounts of data from other domains to enlarge vocabulary coverage and improve n-gram events estimation. The characteristics of telehealth project requires the language model 1) to provide a good coverage of medical terms that may appear in doctor-patient conversation and 2) to effectively deal with frequently-occurring unseen events (word sequence) involving medical terms and driven by different speech styles. Toward this goal, we tried many possible approaches to train word trigram language models, class trigram language models based on different class definitions obtained using various word clustering approaches, and to explore different interpolating methods. Also during the exploring course for language modeling, we found and investigated an intuitive but effective approach that combines class trigram language model trained by using in-domain text data with word trigram language model trained from out-of-domain data, and compared its performance in terms of perplexity with those of a baseline word trigram language model trained from interpolation of in-domain and out-of-domain word trigram models. Details of these approaches and language model performances in term of perplexity are covered in following chapters.

## 1.4. Thesis Organization

The remainder of this thesis is organized as follows.

We begin in chapter 2 with basics of language modeling, including word n-gram language models, class-based n-gram models, language model smoothing and interpolation methods and evaluation metrics usually used for language model performances.

Chapter 3 will discuss several language modeling methods we tried in order to get a robust language model, and explore our own way of interpolating word and class n-gram language models that yield to a satisfying performance in perplexity and recognition results.

Chapter 4 focuses on word clustering methods were used in our experiments. Our own method of defining meaningful semantic word classes is also described in this chapter.

In chapter 5, target dataset and other datasets from different resources are compared in speaking style and topic. Latent semantic analysis is performed to see the correlations between various datasets, aiming to find separate in-domain and out-of-domain datasets that will be used differently in language model training.

Chapter 6 shows experiment settings and all experiments conducted toward our goal to find an ideal language model for the telehealth task. Comparisons between language modeling techniques, between word clustering methods are also revealed in this chapter. The results show that our approach to combining word trigram language model training from in-domain data and class trigram language model trained from out-of-domain data outperformed word trigram language model and pure class trigram language model in terms of perplexity and word error rate.

Chapter 7 summarizing the approaches we tried in language modeling for speech recognition task in telehealth and draw a conclusion that for a medical domain problem involving spontaneous conversational language, a mixture language model “in-domain class and out-of-domain word language model” on a specific medical term definition did give satisfying results and help improve recognition accuracy.

# Chapter 2

# Basics of Language Modeling

Statistic language modeling is a task for assigning a probability to a given sequence of words. It is widely used in many applications. For an ASR system, a good language model generally helps in two ways: 1) reducing search place for word sequence prediction and 2) improve recognition performance by providing context information.

For ASR system, the goal is to find a most likely sequence of words that match an utterance, and give the recognition results in text. A main difficulty to this problem is that there may be innumerable candidate words for certain word positions. For an ASR task involving a vocabulary of 45K words, the number of possible word sequences of a word string with 5 words is $45{,}000^\wedge 5$. This is a huge space for searching, and in practice it is impossible to search all possible candidate answers. Language models, providing useful probabilities estimated based on proper training text corpora can effectively reduce the search space by ignoring unlikely candidates. This is easy to understand through the following example. For a word sequence such as:

"She is drinking a glass of ___."

Given such a word history, the possible following words may be "water", "milk" or "juice" etc. The words like "elephant", "car" and "computer" are presumably unlikely, and ignored with the help of a language model.

On the other hand, language models can provide context information for ASR to eliminate ambiguity. For instance, to choose a word for a certain position in a sentence from multiple candidates with same pronunciation, if no context information is given by language model, incorrect recognition may occur. The following example may explain this clearly. Words “blue” and “blew” can hardly be distinguished from each other if no left and right contexts are considered. However, in the context such as,

“The wind ___ away the leaves,”

the proper one of the two candidate words that can fit in this sentence is the verb “blew.” With the help of a language model that gives a comparatively higher possibility to the trigram “a wind blew” than to “a wind blue”, we can decide the word in the blanked position easily.

In our telehealth project, we used the most popular n-gram models for language modeling, which models language statistically by estimating probability for parameters in forms of unigram (a sequence consisting of one word), bigram (a sequence consisting of two words), trigram (a sequence consisting of three words) and so on.

## 2.1 Word N-gram Language Models

Word n-gram language model is the most popular statistic language model in use. Given a certain word sequence, the probability of it occurring as a sentence is denoted as $P(W)$, which can be calculated in the following way:

$$\begin{aligned} P(W) &= P(w_1, w_2, ..., w_n) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2)\cdots P(w_n \mid w_1, w_2, ..., w_{n-1}) \qquad (2) \\ &= \prod_{i=1}^{n} P(w_i \mid w_1, w_2, ..., w_{i-1}) \end{aligned}$$

where $P(w_i \mid w_1, w_2, ..., w_{i-1})$ is the probability that word $w_i$ will follow the word sequence $w_1, w_2, ...w_{i-1}$, and the sequence $w_1, w_2, ...w_{i-1}$ is the word history presented previously. If we take entire history into consideration, for a vocabulary size *V*, there would be $V^i$ possible events to estimate. Using the chain rule and order-n Markov assumption, we assume a word appearance only depends on *n* previous words. Therefore, if we define a language model under assumption that the appearance of a possible word depends only on its previous two words, we get a trigram language model, with probability parameters estimated for a word group in form of $P(w_i \mid w_{i-2}, w_{i-1})$. In the same way, if it is a bigram language model, the word occurrence would be thought rely only on previous one word. The parameter estimated is like $P(w_i \mid w_{i-1})$. Although many different n-gram models are used in various applications, the most widely used n-gram language models are trigram ones, which are found to be particularly powerful, since usually words have strong dependence on the previous two words. A trigram language model estimates word sequence probability in the following way:

$$\begin{aligned} P(W) &= P(w_1, w_2, ..., w_n) \\ &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1, w_2) \cdots P(w_n \mid w_{n-2}, w_{n-1}) \qquad (3) \\ &= P(w_1)P(w_2 \mid w_1)\prod_{i=3}^{n} P(w_i \mid w_{i-2}, w_{i-1}) \end{aligned}$$

In our telehealth task, all language models we trained are trigram language models.

In order to estimate $P(w_i \mid w_{i-2}, w_{i-1})$, an adequate text data set is needed for parameter probability estimation. This kind of text set is called as a training corpus. And the parameter value is simply estimated by employing maximum likelihood (ML) estimation. To build the model, we calculate word trigram model parameters in the following way:

For trigrams, using ML, a parameter value is estimated as:

$$P(w_i \mid w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \tag{4}$$

For bigrams, it is like:

$$P(w_i \mid w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})} \tag{5}$$

And for unigrams, we estimate a parameter in a way like:

$$P(w_i) = \frac{C(w_i)}{\sum_{w_i' \in V} C(w_i')} \tag{6}$$

In the equations, $C(w_{i-k},..., w_i)$ is the number of appearances, or in another word, frequency of the word sequence $w_{i-k},..., w_i$ in the training dataset. And $V$ in the formula represents the vocabulary.

## 2.2 Class N-gram Language Models

We know that a word can have similar meaning to some other words, or share semantic similarities with some other words. We call such words with similar meanings to a certain word as synonyms of it, such as – words “utensil” and “instrument” to “tool”. And sometimes a word is found functionally similar to other words, that is, they may have similar grammatical function in sentences, like “red”, “blue” and “yellow” which are used as adjectives, modifying a noun and giving information about the color of the thing. This property of words makes it possible to make more reasonable predictions for n-grams that we have not previously seen, by grouping similar words into classes, and assuming words in the same class may occur in the same context or history. Once the words are successfully classified, it can be another effective way to handle the data scarcity problem.

A word can be uniquely assigned to a certain class only, i.e. a one-to-one word-class mapping, or assigned to more than one class, i.e. a one-to-many word-class mapping. A word class can have only one word as its member, or have more than one members, depending on different application requirements, classification algorithms and domain constraints. Once words are classified into corresponding classes, various approaches could be chosen for further training a class-based language model.

The basic class-based language model is proposed in [2]. Supposing the probability of a word belonging to a certain class is $P(w_i \mid c_i)$, and the probability of the certain class following a history involving previous two words is $P(c_i \mid c_{i-2,} c_{i-1})$, giving a word that is uniquely mapped to only one class, as an example, a trigram class model computed based on previous 2 classes is formulated as:

$$P(w_i \mid c_{i-2,}c_{i-1}) = P(w_i \mid c_i)P(c_i \mid c_{i-2,}c_{i-1}) \qquad (7)$$

and $P(w_i \mid c_i)$ is estimated by:

$$P(w_i \mid c_i) = \frac{C(w_i)}{C(c_i)} \qquad (8)$$

$C(c_i)$ is the total number of all the words in the training set that have been categorized to class $c_i$. And $P(c_i \mid c_{i-2,}c_{i-1})$ is estimated as:

$$P(c_i \mid c_{i-2,}c_{i-1}) = \frac{C(c_{i-2,}c_{i-1}, c_i)}{\sum_c C(c_{i-2,}c_{i-1}, c)} \qquad (9)$$

$\sum_c C(c_{i-2,}c_{i-1}, c)$ is the summation of the counts of class trigrams starting with class $c_{i-2}, c_{i-1}$.

If the words are mapped to more than one class types, a sentence that is estimated using a more general class trigram model can be expressed as:

$$P(W) = \sum_{c_1 \ldots c_n} P(w_i \mid c_i)P(c_i \mid c_{i-2,}c_{i-1}) \qquad (10)$$

Some other class-based n-gram language modeling techniques are also proposed by some research groups [4, 13, 15]. We will see a couple of them in the later chapters.

## 2.3 N-gram Language Model Smoothing

N-gram language model is easy to build. Given a dataset as the training corpus, what need to do is to count the word or word sequence frequencies. One problem that can appear in use of a language model, or in testing of the language model, is that we may not reliably estimate the probabilities for some n-grams, because those certain n-gram events were not seen in the training corpus which might be too small in size and therefore does not cover as many n-gram events as needed. However, due to the sparseness of data, this kind of unseen events can very possibly happen. Even when the training set is large with millions of words and with a vocabulary size being more than thirty thousands, it is still very possible for some word successions being poorly observed or unseen. Taking Switchboard and Broadcasting News datasets we used for training as example, although each dataset contains 2.9 million words, in neither of them could we find a quite frequently used word sequence "*take any medications*". In such a case, count of the certain trigram event $C(w_{i-2}, w_{i-2}, w_i)$ might be zero, thus, the probability of such trigram to happen $P(w_i \mid w_{i-2}, w_{i-1})$ will be estimated as zero. Therefore, the model would give zero to the probability of a word sequence, $P(W) = P(w_1, w_2, ..., w_n) = 0$ once there is an unseen n-gram word string existing in the sequence, even if it is grammatically correct or logically acceptable, and highly possible to appear in practice. This problem will be critical to our ASR application. Language model smoothing is needed to solve this problem, to make language model more robust.

The core issue of smoothing is to assign a nonzero probability to word strings, to reduce hard errors in recognition process. And this is done generally by reserving some small probability mass from the relative frequency calculation of the seen events (n-gram word strings) in the training data and assign this subtracted value to unseen events. This produces more robust probabilities for unseen events, therefore improves the recognition accuracy, while not affecting too much the estimation of seen events.

Several smoothing techniques have been proposed during the last decades. Some of them are described as linear interpolation smoothing, which interpolates higher and lower-order n-gram models in the following way:

$$\begin{aligned}&P'(w_i \mid w_{i-n+1},...,w_{i-1})\\&=\lambda_{w_{i-n+1}\dots w_{i-1}} P(w_i \mid w_{i-n+1},...,w_{i-1}) + (1-\lambda_{w_{i-n+1}\dots w_{i-1}}) P'(w_i \mid w_{i-n+2},...,w_{i-1}), \quad (11)\\&\qquad\qquad\text{with} \quad 0 < \lambda_{w_{i-n+1}\dots w_{i-1}} < 1\end{aligned}$$

The nth-order gram is interpolated by the nth-order maximum likelihood model and the (n-1)th-order smoothed model. This smoothed model is defined recursively, with the 1$^{st}$-order gram (unigram) being the maximum likelihood model.

Some other popular smoothing techniques are defined in form below:

$$\begin{aligned}&P'(w_i \mid w_{i-n+1},...,w_{i-1})\\&=\begin{cases} P(w_i \mid w_{i-n+1},...,w_{i-1}), \text{if } C(w_{i-n+1},...,w_i) > 0 \\ \alpha(w_{i-n+1},...,w_{i-1}) P'(w_i \mid w_{i-n+2},...,w_{i-1}), \text{if } C(w_{i-n+1},...,w_i) = 0 \end{cases} \quad (12)\end{aligned}$$

In this way, if the n-gram is seen in training data, the maximum likelihood estimated probability will be used. Otherwise, we back off to the smoothed lower-order model, that is, estimate the n-gram probability using the (n-1)th-order gram distribution $P'(w_i \mid w_{i-n+2},...,w_{i-1})$ with a scaling factor $\alpha(w_{i-n+1},...,w_{i-1})$ which is used to make the conditional distribution sum to one. Smoothed language models obtained in this way are also called backoff models. In our ASR project, two popular backoff smoothing algorithms were used, Katz smoothing and Kneser-Ney smoothing. In the following sections we will view the two smoothing techniques simply.

### 2.3.1 Katz Smoothing:

Katz smoothing [7] follows the intuitions of the Good-Turing estimation. Good-Turing method partitions n-grams into groups according to the frequency. And for any n-grams that occur *r* times in the text corpus, a "discounting" procedure will modify the count *r* and replace it by $r^*$, which is got by:

$$r^* = (r+1)\frac{n_{r+1}}{n_r} \qquad (13)$$

in which $n_r$ is the number of n-grams occurring exactly *r* times. Then the probability of an n-gram that occurs r times in the training corpus will be:

$$P^* = \frac{r^*}{N} \qquad (14)$$

where *N* is the total number of words in the test corpus, that is,

$$N = \sum_r r n_r \tag{15}$$

Katz smoothing extends Good-Turing estimate by adding the combination of higher-order models with lower-order models. Using bigram language model smoothing problem as example, it suggests estimating bigram counts as:

$$C^*(w_{i-1}, w_i) = \begin{cases} d_r r, & \text{if } r > 0 \\ \alpha(w_{i-1}) P(w_i), & \text{if } r = 0 \end{cases} \tag{16}$$

where $d_r$ is called discount coefficient that approximately equals to $r^*/r$. In this way, the counts for all n-grams with counts larger than zero are discounted according to the discount coefficient, and the subtracted counts are redistributed among the zero-count n-grams according to the next (n-1)th-order grams. To make sure the total number of counts remains unchanged in the distribution, i.e., $\sum_{w_i} C^*(w_{i-n+1}, ...w_i) = \sum_{w_i} C(w_{i-n+1}, ...w_i)$, a factor $\alpha$ is needed:

$$\begin{aligned} \alpha(w_{i-1}) &= \frac{1 - \sum_{w_i : C(w_{i-1}, w_i) > 0} P^*(w_i \mid w_{i-1})}{\sum_{w_i : C(w_{i-1}, w_i) = 0} P(w_i)} \\ &= \frac{1 - \sum_{w_i : C(w_{i-1}, w_i) > 0} P^*(w_i \mid w_{i-1})}{1 - \sum_{w_i : C(w_{i-1}, w_i) > 0} P(w_i)} \end{aligned} \tag{17}$$

$P^*(w_i \mid w_{i-1})$ is calculated from the modified counts:

$$P^*(w_i \mid w_{i-1}) = \frac{C^*(w_{i-1}, w_i)}{\sum_{w_k} C^*(w_{i-1}, w_k)} \qquad (18)$$

According to Katz, large counts, like in a range of five to eight, are reliable, and these n-gram counts are not to be discounted. Counts lower than a certain threshold, *k*, are considered to be replaced using the coefficient $d_r$. $d_r$ is given by:

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \qquad (19)$$

So, Katz backoff model can be summarized as:

$$P_{Katz}(w_i \mid w_{i-n+1},...w_{i-1}) = \begin{cases} C(w_{i-n+1},...w_i)/C(w_{i-n+1},...w_{i-1}), & \text{if } r \geq k \\ d_r C(w_{i-n+1},...w_i)/C(w_{i-n+1},...w_{i-1}), & \text{if } k > r > 0 \\ \alpha(w_{i-n+1},...w_{i-1}) P_{Katz}(w_i \mid w_{i-n+2},..., w_{i-1}), & \text{if } r = 0 \end{cases} \quad (20)$$

$P_{Katz}(w_i \mid w_{i-n+1},...w_{i-1})$ is recursive defined, and the recursion is ended at the unigram model got by maximum likelihood estimation.

### 2.3.2 Kneser-Ney Smoothing:

This algorithm [8] is based on the absolute discounting technique, which is different from the Good-Turing discount. Considering this situation that in a text corpus, there are words with high frequencies, however, only following a certain history, like the word “Angeles" which only occurs after word “Los”. Since the number of occurrences of this word is big, for example, 16 times in the whole training set, the probability of word “Angeles" estimated from the training set can be very high. To estimate a bigram including the word “Angeles” with a novel history, Katz model will assign a relatively high probability to occurrence of an unseen bigram by backing off bigram estimation to unigram of “Angeles”, although the bigram might not be possible to exist in any language circumstances. This is a potential problem of over-estimation. Different from Katz smoothing technique, according to Kneser-Ney’s opinion, the unigram probability should not be proportional to the number of occurrences of a single word in the training dataset. Instead, it should be proportional to the number of different words that it follows. Thus, the probability of occurrence of a certain word is estimated as:

$$P(w_i) = C(\bullet w_i) / \sum_{w_i} C(\bullet w_i) \tag{21}$$

$C(\bullet w_i)$ means the number of different words that precede $w_i$. And, the probabilities for higher-order gram models are estimated as:

$$\begin{aligned}&P_{KN}(w_i \mid w_{i-n+1},\dots w_{i-1})\\&=\frac{\max\{C(w_{i-n+1},\dots w_i)-D,0\}}{\sum_{w_i} C(w_{i-n+1},\dots w_i)}+(1-\lambda_{w_{i-n+1},\dots w_{i-1}})P_{KN}(w_i \mid w_{i-n+2},\dots w_{i-1})\end{aligned} \quad (22)$$

To make the distribution sum to 1,

$$1-\lambda_{w_{i-n+1},\dots w_{i-1}}=\frac{D}{\sum_{w_i} C(w_{i-n+1}\dots w_i)}C(w_{i-n+1}\dots w_{i-1}\bullet) \quad (23)$$

where $C(w_{i-n+1}\dots w_{i-1}\bullet)$ is the number of different words following the history $w_{i-n+1}\dots w_{i-1}$, and D is a fixed discount. In this way, not only the unseen nth-order grams with zero counts but all grams are interpolated with the lower-order distribution.

Both algorithms were used in the experiments for language modeling in our ASR project. Experiments and results will be covered in later chapters.

## 2.4 Language Model Interpolation

If for a task, enough in-domain data is acquired or easy to collect, it will be very helpful for training a satisfying language model. However, for many tasks, especially tasks involving conversational speech in certain specialty domains, this kind of ideal training material is hard to acquire. Data for this kind of task, including telehealth project, is costly to produce because directly related data got from transcripts of conversational speech with same or related topic is limited. Thus, for this kind of task the amount of training data is usually insufficient. Under such circumstances, language models trained using some datasets about different topics, or gotten from datasets with different text styles might be used as supplements and combined with the one trained from the target text data. Among these datasets from different data sources, the ones sharing similar topic with the target data are mainly used for vocabulary word coverage and the ones with similar styles to the target data are used for better n-gram event estimation.

The simplest and most general way to combine several language models is linear interpolation, in which, weights are given to those language models needed to be combined according to their relevancy, similarity to the target application data, or their function in term of covering vocabulary or modeling style. The following formula shows the idea of general linear interpolation of language models using trigram language models as example:

$$P(w_i \mid w_{i-2}, w_{i-1}) = \sum_{j=1}^{k} \lambda_j P_j(w_i \mid w_{i-2}, w_{i-1}) \qquad (24)$$

Here, *k* is the number of language models being interpolated, and $\lambda_j$ is the weight assigned to the *jth* language model.

## 2.5 Language Model Evaluating Metrics:

### 2.5.1 Word error rate (WER)

The most common metric for evaluating automatic speech recognition (ASR) is word error rate (WER), which is simply a ratio of the number of incorrectly recognized words in ASR output to the total number of words (*N*) in the reference text of the test set. It is also used as a metric for language model performance evaluation. Recognition errors include insertions (*I*), deletions (*D*) and substitutions (*S*). Then the word error rate is defined as follow:

$$WER = \frac{D + I + S}{N} \times 100\% \qquad (25)$$

### 2.5.2. Test Set Perplexity

WER requires the evaluation of a speech recognition system, which is sometimes computational costly. Another popular and simpler way for testing the performance of a language model is to measure the average probability assigned by a language model to each word given its linguistic context in the test set *W*. This is a measure related to cross-entropy called as test set perplexity [5]

$$PPL(W) = 2^{H(W)} = 2^{(-\frac{1}{N}\log_2 P(W))} = P(W)^{(-1/N)} \qquad (26)$$

It shows how the language model fits the training data and can be roughly interpreted as the geometric mean of the branching factor of the test text when presented to the language model, that is, the average number of word choices for each word position. If the test set perplexity for a language model is 500, it means that there are 500 possible choices on average for every word position. The higher the perplexity is the more branches may follow a word history, the more possible words needing to be predicted statistically. Contrarily, lower perplexity will generally have less confusion in recognition. So generally speaking, lower perplexity correlates with better recognition performance. However, it is not always true. There have been quite a few reports that showed language models with lower perplexity led to worse ASR WER. Therefore, in our experiments, we used perplexity as a main language model metric while using ASR WER to verify the evaluation results as well. A consistency between the evaluation results may be seen using these two measurements.

# Chapter 3

# Word-Class mixture language Models

## 3.1 Hierarchical backoff model

The hierarchical backoff language model [15] hierarchically clusters vocabulary words into a tree like structure, with each node of the word tree being a cluster. Each cluster contains some sub clusters and words belonging to those sub clusters. The bottom level of the hierarchical structure, that is, the leaves, represent individual words. The closer the node to the root, the more general concept the cluster represented by the node contains.

The idea of this algorithm is to take advantage of both word n-gram for frequent events and the predictive power of class n-grams for unseen event. Concept generalization ability and word specificity are balanced and used to estimate probabilities of low-frequency or unseen events. Using a trigram event as example, if a trigram is seen in the training corpus, the probability of this event is estimated by word trigram model got from discounted maximum likelihood estimation, as mentioned in previous section about Good-Turing discounting. If it is an unseen event, a class-based trigram model will be used for estimation in form of $P(w_i \mid C_{i-2}^{j}, w_{i-1})$, in which $C_i^j$ means the *jth* level class the word $w_i$ belonging to. The idea is formulated as:

$$P(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \tilde{P}(w_i \mid w_{i-2}, w_{i-1}) = d_{N(w_{i-1},\ldots w_i)} \dfrac{N(w_{i-2},\ldots w_i)}{N(w_{i-2}, w_{i-1})} & \text{if } N(w_{i-2},\ldots w_i) > 0 \\ \alpha(w_{i-2}, w_{i-1}) P(w_i \mid C_{i-2}^{1}, w_{i-1}) & \text{otherwise} \end{cases} \qquad (27)$$

And if $P(w_i \mid C_{i-2}^{1}, w_{i-1})$ is not seen, the parent level of class $C_i^1$, which is $C_i^2$ is checked. If the trigram class probability in parent level $C_i^2$ is still unseen, a more general context will be examined for the trigram, which is one level up in the cluster tree. A more general context is used recursively until the root of the tree is met, in which case, the word trigram back-offs to a word bigram probability. The whole process in formula is described as:

$$P(w_i \mid C_{i-2}^{j}, w_{i-1}) = \begin{cases} \tilde{P}(w_i \mid C_{i-2}^{j}, w_{i-1}) & \text{if } N(C_{i-2}^{j}, w_{i-1}, w_i) > 0 \\ \alpha(C_{i-2}^{j}, w_{i-1}) P(w_i \mid C_{i-2}^{j+1}, w_{i-1}) & \text{if } N(C_{i-2}^{j}, w_{i-1}, w_i) = 0 \text{ and } C_{i-2}^{j+1} \text{ is not the root} \\ \alpha(C_{i-2}^{j}, w_{i-1}) P(w_i \mid w_{i-1}) & \text{if } N(C_{i-2}^{j}, w_{i-1}, w_i) = 0 \text{ and } C_{i-2}^{j+1} \text{ is the root} \end{cases} \qquad (28)$$

where $\alpha(\bullet)$ is a factor that guarantees the probabilities sum to 1.

$$\alpha(C_{i-2}^{j}, w_{i-1}) = \begin{cases} \dfrac{1 - \sum_{w_i : N(C_{i-2}^{j}, w_{i-1}, w_i) > 0} \tilde{P}(w_i \mid C_{i-2}^{j}, w_{i-1})}{1 - \sum_{w_i : N(C_{i-2}^{j}, w_{i-1}, w_i) > 0} \tilde{P}(w_i \mid w_{i-1})} & \text{if } C_{i-2}^{j+1} \text{ is the root} \\ \dfrac{1 - \sum_{w_i : N(C_{i-2}^{j}, w_{i-1}, w_i) > 0} \tilde{P}(w_i \mid C_{i-2}^{j}, w_{i-1})}{1 - \sum_{w_i : N(C_{i-2}^{j}, w_{i-1}, w_i) > 0} \tilde{P}(w_i \mid C_{i-2}^{j+1}, w_{i-1})} & \text{otherwise} \end{cases} \qquad (29)$$

The corresponding algorithm to construct a hierarchical word classes will be talked about in the next chapter in details.

## 3.2 Word-Class N-gram Mixture Language Model

Word n-gram language models have been successfully used in many speech recognition systems in which large amounts of text data are available for language model training. However, low-frequency or unseen events cannot be reliably estimated in many applications by using word n-gram language models. Such applications include specific topic tasks and conversational speech style tasks, where in-domain text materials are usually limited in amount and costly to produce. Consider an example from our telehealth task. Word sequences with medical words, such as, “take the amitriptyline”, only appear in telehealth dataset once. The probability the system predicts this word sequence will be very low if a word n-gram language model is used, even if this sequence is quite possible to appear in practice.

However, if we group words with similar semantic meaning together and make use the fact that semantically resembled words can share same word position in sentences, we can predict a word through its word class. For example, based on its semantic property, we classify the word “amitriptyline” into a group consisting of medicines and pharmacologically agents, given a class name as “MDCN”. In the same class group are words such as “amoxicillin,” “cimetidine,” “digoxin” and so on. If we assign these words with its class name “MDCN”, and count word sequence like “take the MDCN” in the dataset, 19 times of such sequences are found in the dataset. Such observation implies a much higher possibility for sequence “take the amitriptyline” to take place, since amitriptyline is a kind of “MDCN”. This is just an example showing how class n-gram language models help estimate low frequency n-gram event. Not only can they work well on low-frequency words, but on unseen events also. Using the same example, suppose a sequence like “take the digoxin” needs to be estimated using the language model, however such a sequence never occurred in training dataset. Using a word n-gram language model, this event might be treated as an event not possible to happen if no smoothing technique is used, or as an event with very low possibility if a smoothing technique is used such as to back off the possibility of the trigram event to the unigram estimate of word “digoxin”. However, using class n-gram model, as long as we know the word class “MDCN” of the word “digoxin”, we can estimate its occurrence by the class, thus a reasonable estimation of probability of “take the digoxin” is gained even it never appears in training set. For such reason mentioned above, class n-gram model is considered to be suitable and useful for some domain-dependent language modeling tasks. And it has been a common understanding that comparing to word n-gram language models, class n-gram language models with good predictive capability on low frequent events and unseen events is powerful for some small vocabulary domain-specific applications with small amounts of training data. However, class n-gram language models are not portable, which means that for different tasks, different word class definitions are needed, and class n-gram models need be retrained. So, for some language modeling task with enough proper training data, word n-gram language models have been reported to perform better than class n-gram language models.

Considering the characteristics of word and class language models, it is desirable to make a mixture model that retains advantages of both class and word n-grams by combining their word prediction. In the telehealth project, the in-domain telehealth (TH) data is still very small in size, with only 119k words. However, the TH dataset contains a large ratio of medical terms - 10.7% of the vocabulary is medical terms. Most of the medical terms appear in TH dataset with very low frequencies. An experiment shows that 68.9% of these terms have frequency lower than 5 times, 57.2% of them appear not more than 3 times and 37.8% appear only once. Word n-gram model may miss many quite possible n-gram events containing medical terms, leading to poor word sequence estimation. A more general class language model will work better under this circumstance. However, for a large-sized out-of-domain corpus containing a few medical terms, class language model could not manifest its superiority. Therefore, an intuitively appealing approach is proposed to interpolate a class language model gained from in-domain data and a word language model trained from out-of-domain datasets, so as to bring advantage of each kind of model into play.

Denote in-domain class n-gram model by *IC* and out-of-domain word n-gram model by *OW*, the proposed interpolation method becomes:

$$\begin{aligned}
&P(w_i \mid w_{i-2}, w_{i-1}) \\
&= \lambda_{IC} P_{IC}(w_i \mid w_{i-2}, w_{i-1}) + (1-\lambda_{IC}) P_{OW}(w_i \mid w_{i-2}, w_{i-1}) \\
&= \lambda_{IC} \sum_{j \in I} \alpha_j P_{IC}^{j}(w_i \mid w_{i-2}, w_{i-1}) + (1-\lambda_{IC}) \sum_{k \in O} \alpha_k P_{OW}^{k}(w_i \mid w_{i-2}, w_{i-1})
\end{aligned} \tag{30}$$

where the *IC* and *OW* models are each a mixture model as well.

In Chapter 6, experimental results will show that the proposed interpolation outperforms both pure word mixture language model and pure class mixture model.

# Chapter 4

# Word Clustering

For class-based language modeling, we need to group words into classes according to their similarity in term of syntactic function or semantic content or both. Good word class definitions will help a lot for class n-gram language modeling, by giving reasonable class definition to words, thus correctly estimating the word occurrence possibilities in a linguistic context. In order to build meaningful word classes, we have experimented on three popular approaches, POS tagging approach, the hierarchical word clustering for hierarchical POS back-off language modeling, and automatic clustering algorithm. Besides, we investigated medical-term based semantic classes focusing on our specific medical domain task and found the latter to be superior to the first three methods through experiments.

## 4.1. POS Tagging Method

Parts of speech are the traditional name for syntactic or grammatical categories of words. Some important parts of speech are *noun* (NN), *verb* (VB) and *adjective* (AD). POS tagging is a task of labeling or tagging each word in a sentence with its appropriate part of speech. An example of POS tagging using Penn Treebank tag set is:

He/PRP got/VBD a/DT headache/NN because/CC of/IN the/DT tension/NN.

In English, many words can function differently in sentences, for example, the word “can” is used as a modal verb in “*You can do it*”, however in a phrase “*a can of tuna*” it performs as a noun. So, the tagging task tries to determine which one of these possible parts of speech is the most likely one for a particular word in a specific sentence. The context the word falling into gives important information for finding a most appropriate tag, so a way to tag words is to look at the tags of other words in the context of the target word. Although the parts of speech of these words may also be ambiguous, the essential observation shows that some part of speech sequences are more common than others. Besides, the word itself gives lots of information about a correct tag too. Thinking about a word that can be noun and verb, say, *flour*, although it could work as a verb in “*flour the pan*”, the frequency for it being used as a verb is quite low than as a noun. Thus, both the tags of other words in the context and the probability of a word belonging to a certain POS class need be considered in order to tag words as correctly as possible.

A POS tagging system is trained from some annotated text corpora, using techniques as Markov Model [10], Statistical Decision Tree [6] or maximum Entropy model [12]. Here we will only briefly talk about Markov Model tagging method. Given a manually tagged text as a training set, we can get probability of tag $t_k$ following $t_j$ using the maximum likelihood estimation based on frequencies as:

$$P(t_k \mid t_j) = \frac{C(t_j, t_k)}{C(t_j)} \qquad (31)$$

And the probability of a word belonging to a certain POS class is obtained via maximum likelihood estimation as:

$$P(w_l \mid t_j) = \frac{C(w_l, t_j)}{C(t_j)} \tag{32}$$

Thus the problem of giving optimal tags to a sentence is:

$$\hat{t}_{1,n} = \underset{t_{1,n}}{\arg\max} P(t_{1,n} \mid w_{1,n}) = \underset{t_{1,n}}{\arg\max} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1}) \tag{33}$$

Making use of a tagging system named “mxpostSoftware” we tagged our training corpus with POS tags. The system was trained on annotated Wall Street Journal corpus, following Penn Treebank tag set [12]. Penn Treebank tag set is one most widely used tag set nowadays, consisting of 45 kinds of tags, and is a simplified version of the Brown tag set [9]. We take use of this tagging system to tag our training data sets using the 45 types of tags, and then made some modification on the tags based on the need on our specific language modeling task for telehealth, whose details will be covered in following chapters and sections.

Since most English words have more than one part-of-speech tags, POS tagging mechanism can assign a word to different word classes. Again, using the word “*flour*” as an example, it can be tagged as NN (noun) or VB (verb). This kind of one-to-many mapping for word classes will complicate the speech decoding process, adding a process for guessing the possible word class history. To simplify this problem, we considered to further group words with the same set of POS tags into one class. For instance, word "*dance*" is tagged by both NN (noun) and VB (verb) after tagging process, since it can work as a noun or a verb in different context. Same set of tags, NN and VB, is given to word "*guarantee*" too. So, in the step of merging tags, these two words are considered to be grouped into one class referenced by "NN_VB". For words being classified into more than 3 POS classes, like "above", which is tagged by RB (adverb), JJ (adjective), IN (preposition) and NN (noun), we do not want to keep all these possible POS tags for the word, because some of them are really rare in use, like NN for "*above*". It does not make much sense to keep the rare ones and make the word hard to be grouped with other words that are actually similar to it in most of cases. Thus, we truncated the tag list given to a certain word by keep only the first 3 POS tags that the word most frequently functions as. By counting the frequency of a certain tag given to a word, we rank the tags in an order based on frequency from high to low. Lower-frequency tags are truncated by cutting off tags falling out of top three positions. And those tags with a frequency extremely lower than that of the top ranked tag are also discarded. If we use the example we mentioned above to explain this idea, the word "above" will be assigned to a class given a name as IN_RB_JJ.

## 4.2 Hierarchical word clustering algorithm

The method proposed in [15] is to construct the hierarchical structure of vocabulary words by recursively merging similar words into groups. The similarity between words is decided by the contextual information. Using bigram context as an example, the discriminative information between two words is estimated as:

$$D(w_i, w_j) = \sum_{v=1}^{V} pl(w_v \mid w_i) \log \frac{pl(w_v \mid w_i)}{pl(w_v \mid w_j)} + \sum_{v=1}^{V} pr(w_v \mid w_i) \log \frac{pr(w_v \mid w_i)}{pr(w_v \mid w_j)} \quad (34)$$

in which $pl(w_v \mid w_i)$ is left hand side bigram probability for word $w_i$, which is the probability of the sequence "$w_v, w_i$" and $pr(w_v \mid w_i)$ is right hand side bigram probability for word $w_i$, which is the probability for sequence "$w_i, w_v$". The goal of partitioning vocabulary words is to find a set of centroids $\{o_c\}$ for clusters $\{O_c\}$ which leads to a minimum global discriminative information:

$$GDI = \sum_{c=1}^{C} \sum_{i \in O_c} D(w_i, o_c) \quad (35)$$

And the centroid $o_c$ of cluster $O_c$ is defined as the mean of all context vectors of words $w_i$ in the cluster.

$$o_c(k \mid O_c) = \frac{1}{v_c} \sum_{i=1}^{v_c} p(k \mid w_i) \quad (36)$$

A context vector of word $w_i$ is defined as $p(w_i) = \{pl\{w_i\}, pr\{w_i\}\}$, where $pl\{w\} = \{p(w_1, w_i), ..., p(w_V, w_i)\}$ and $pr\{w\} = \{p(w_i, w_1), ..., p(w_i, w_V)\}$. In equation (36), $p(k \mid w_i)$ is a simplified notation for both left hand side and right hand side bigram probability for word $w_i$ in $O_c$.

Thus, the clustering algorithm consists of the following steps:

- Initial step: find the global centroid of the data space, and set the *C* words closest to the global centroid as the centroids of *C* clusters.
- Step 2: for each $w_i$ find the closest cluster $O_c$ by calculating $D(w_i, o_c)$, and assign it to $O_c$.
- Step 3: update clusters
- Step 4: **if** *GDI >t*, with t being a threshold, then go to step 2
- Step 5: **if** there is any *O* whose number of members is less than a threshold *k*, then $(C \leftarrow C - 1)$, repeat from initial step. **Else**, stop.

Once C classes have been defined for one level of the hierarchical tree, further clustering will be done in each cluster, to grow the cluster tree.

In our experiment about POS hierarchical backoff language modeling, we follow the technique mentioned in section 3.1., however, considering our specific medical domain task, we defined the hierarchical levels of word vocabulary in our own way.

Words in vocabulary are tagged with the help of POS tagging system. To construct hierarchical levels for a vocabulary, we further group words with most related or functionally highly similar tags into a more general class and give the class a new tag. For example, words belonging to NNS (plural noun), NNP (proper noun) and NNPS (plural proper noun) are grouped into a more general class named as "noun" represented by NN. In the table bellow, we can see more examples of word class hierarchies. Moreover, considering the topic of the telehealth task, we added a class for medical terms at the second level named as NNM, and further classified nouns in this NNM class into four subclasses, which are, disease (D), medicine (M), body structure (S) and meditation techniques (T). Also, we added one category for medically related adjectives and gave a class name as JJM. It is also in the second level and is a subclass of JJ (adjective). Table 1 shows three levels of the word class hierarchy with examples. As we can see, some of the classes do not have subclasses, while some of them are further splitted into smaller classes. It is all depends on word class properties.

*Table 1* Basic POS Hierarchical Structure Defined for POS Language Modeling on telehealth

| Root | Level 1 | Level 2 | Level 3 | example |
|---|---|---|---|---|
| | CC (conjunction) | | | and, or |
| | CD (number) | | | one, two |
| | DT (determiner) | DTC (common determiner) | | the, a |
| | | WDT (question determiner) | | which, whatever |
| | | WP$ (possessive determiner) | | whose |
| | | PRP$ (possessive determiner) | | their, your |
| | FW (foreign word) | | | donde |
| | IN (preposition) | | | in, on |
| | JJ (adjective) | JJC (common adjective) | | red, good |
| | | JJR (comparative adjective) | | better, happier |

| | | | | |
|---|---|---|---|---|
| | | JJS (superlative adjective) | | best, happiest |
| | | JJM (medical domain adjective) | | dizzy, numb |
| | MD (model) | | | would, must |
| | NN (noun) | NNC (common noun) | | bank, tree |
| | | NNCS (plural noun) | | books, cats |
| | | NNP (proper noun) | | Missouri, Jackson |
| | | NNPS (plural proper noun) | | Australians, Methodists |
| | | NNM (medical related noun) | D (disease) | anoxia, emphysema |
| | | | M (medicine) | emetin, metformin |
| | | | S (body structure) | elbow, neck |
| | | | T (curing techniques) | intubation, phototherapy |
| | POS (possessive) | | | Jack’s, Emily’s |
| | PRP (pronoun) | PRPF (reflexive pronoun) | | himself, ourselves |
| | | PRPO (object pronoun) | | him, us |
| | | PRPS (subjective pronoun) | | I, we |
| | | WP (question pronoun) | | who, whoever |
| | RB (adverb) | RBC (common adverb) | | fast |
| | | RBR (comparative adverb) | | faster |
| | | RBS (superlative adverb) | | fastest |
| | | WRB (question adverb) | | when, how |
| | TO (to) | | | to |
| | UH (filled pause) | | | um, huh |
| | UNKN (unknown token) | | | |
| | VB (verb) | VBC (verb, present tense) | | read |
| | | VBD (verb, past tense) | | wrote |

| | | | | |
|---|---|---|---|---|
| | | VBG (verb, present participle) | | reading |
| | | VBN (verb, past participle) | | written |
| | | VBZ (verb, -s) | | reads |

The POS hierarchical backoff language models being adopted are based on a class definition that maps each individual word to only one class. For words with multiple classes, we redefined the classes using the same method we explained in section 4.1., i.e. assigning a word using the merged tag got from merging the top ranked tags assigned to it. To define a multiple level tags for a word, we followed the 3 level hierarchy mentioned above. For instance, if a word is assigned to a class NNC_VBD, its parent level class will be assigned as NN_VB with NN and VB being the parent level classes for NNC and VBD respectively.

## 4.3 Automatic Word Clustering Algorithm

This is also recognized as Brown's algorithm [2], which attempts to cluster words into *n* clusters using a greedy clustering method. The idea is to assign each word to a distinct class initially. Then at each merging step, two clusters for which the loss in average mutual information is least are merged to a larger class. The average mutual information at a certain merging step, which is a summation of all pair wise information, is defined as:

$$I = \sum_{l,m} p(l,m) \log \frac{p(l,m)}{pl(l)\, pr(m)} \qquad (37)$$

where

$$\begin{aligned} pl(l) &= \textstyle\sum_m p(l,m) \\ pr(m) &= \textstyle\sum_l p(l,m) \\ p(l,m) &= \Pr(C(l),C(m)) \end{aligned} \tag{38}$$

In the equations (37) and (38), *l* and *m* are words in class $C(l)$ and $C(m)$ respectively. $\Pr(C(l),C(m))$ is the probability that a word in class $C(m)$ follows a word in class $C(l)$. If we use the notation *i+j* to represent the cluster obtained by merging classes $C(i)$ and $C(j)$, the average mutual information remaining after merging the two classes $C(i)$ and $C(j)$ is then:

$$\begin{aligned} I(i,j) = I - s(i) - s(j) + q(i,j) + q(j,i) + q(i+j,i+j) \\ + \sum_{l \neq i,j} q(l,i+j) + \sum_{m \neq i,j} q(i+j,m) \end{aligned} \tag{39}$$

where

$$q(i,j) = p(i,j)\log\frac{p(i,j)}{pl(i)\,pr(j)} \tag{40}$$

$$s(i) = \sum_l q(l,i) + \sum_m q(i,m) - q(i,i) \tag{41}$$

and for $q(i+j,m)$,

$$p(i+j,m) = p(i,m) + p(j,m) \tag{42}$$

By comparing all cluster pairs, we find the two clusters which lead to a loss in average mutual information $L(i, j) \equiv I - I(i, j)$ the least, and merge them into a larger cluster. The iterating process ends when the number of clusters becomes *n*. Further more, according to Brown's finding, the average mutual information can be made larger by moving some words from one class to another. So, after a set of classes have been derived, the algorithm cycles through the vocabulary moving each word to the class for which the resulting partition has the greatest average mutual information. When average mutual information could be not be further gained by any potential reassignment of a word, the whole clustering process stops. This algorithm is an $O(n^3)$ algorithm, which is not effective in terms of processing time.

To see how well the clustering algorithm will work on our language modeling task and to see how the numbers of classes defined would affect the performances of the language models, in which we attempted several class definitions.

## 4.4 Medical term classification

Based on the characteristics of telehealth tasks on a medical domain, we investigated a method of grouping words into semantic classes for only medical related terms, instead of all words from the vocabulary. Currently, our telehealth project is concerned with the medical specialty domain of neuropsychology. Neuropsychology is the study of cognitive (language, memory, etc.) and behavioral symptoms that occur in people with brain disorders. This branch of psychology covers many medical areas involving a large amount of medical terms cross over the fields. Because of this, ASR in telehealth project chose data from this area for training aiming to train models that have reasonable medical term coverage. The task characteristic determines that the application involves quite a large number of medical terms like “neurotransmitter,” “sulfonylurea,” etc. To make word classes more meaningful to this particular medical domain problem and to avoid overly smoothing data, we propose to group together only words by medical semantic categories, such as the names of medicines, diseases and therapeutic techniques. In addition, classes such as digits, people’s names are also categorized. The rest words falling outside of these categories stand by themselves as singleton word classes. Table 2 shows the defined classes. Language model trained based on the proposed word class definition surpasses the POS and automatic clustering based word classes in performance of perplexity.

Our consideration on grouping only semantically similar words, especially the medically-related words together into classes has two aspects. First, telehealth data is in a medical domain, which consists of large quantity of medical words. However, these words themselves appear only a couple of times or less in the small telehealth dataset. Therefore, to estimate their possibility of appearance, we need to group them into corresponding classes, and use context information of the class to help estimating word occurrence probability. Second, these words with high resemblance in meaning are usually found falling into same context. For a word sequence, even though we never saw such a string in training data we can assume that this event is possible if similar words were seen in such a context.

*Table 2* Word classes defined for telehealth and examples

| Description | Example | # distinct words |
|---|---|---|
| Disease | acalculia, paraplegia | 382 |
| Medicine | phenobarbital, precipitant | 525 |
| Human Body & Organs | follicle, capsule | 138 |
| Meditation Method | intubation, phototherapy | 91 |
| Medical Equipment & Facility | sigmoidoscopy, inhalator | 24 |
| Symptom(noun) | numb, stiffness | 37 |
| Symptom(adjective) | dizzy, drowsy | 6 |
| Medical &Chemical Object | peroxide, triglyceride | 66 |
| Profession Name | ophthalmologist, oncologist | 65 |
| Person Name | Garrett, Lewis | 35 |
| Number | One, two | 33 |

# Chapter 5

# In-domain and Out-of-domain Data Analysis

Data directly gained from telehealth (TH) dataset is insufficient for reliable language model training. Therefore, large amounts of data from other sources are needed for better estimation of n-gram parameters and vocabulary coverage. For proper utilization of the heterogeneous datasets, we examine target application data and consider proper complementary data sources that match or partially match the target task in speaking style and/or content topic.

## 5.1 Speech style

The TH dataset is characterized by spontaneous conversational speech between doctors and patients. The speech style can be categorized as an unplanned dialogue. The transcribed data is full of dialogic text, such as sentences beginning with “*ok*”, “*so*”, “*yeah/yep/ya*” and “*well*”. In the dataset, we found 1389 sentences out of totally 10759 sentences beginning with word “*ok*”, 437 sentences beginning with “*so*”, 117 sentences beginning with “*yeah*” or “*yep*” or “*ya*” and 148 sentences beginning with “*well*”. Besides, colloquial words like “*wanna*” and “*gonna*” also appear in the dataset with “*wanna*” counted 141 times, and “*gonna*” for 107 times. These kinds of phrases or word strings appear with very high frequency in the whole dataset. Focusing on this characteristic, conversational datasets are chosen that share similar speech styles. Such datasets include Switchboard (SW) dataset which is transcribed from spontaneous telephone conversational speech and Broadcasting News (BN) dataset which includes spontaneous conversational speech from talk shows as well as read speech in the form of news and voice-overs. Both of these datasets we used for training contains 2.9 million words. In addition, Callhome (CH) dataset of telephone conversations is also used involving 0.2 million words.

## 5.2 Topic/Domain

For the telehealth task, medical word coverage is critical. To cover possible events involving medical terms, we need more datasets that match the telehealth domain. For this purpose we acquired a dataset that comes from conversational speech transcripts in telehealth but on topic of dermatology, referred to as TH-D. A slight speech style difference exists between these two datasets, with TH-D data being simple question-answer style which is short in length, while TH data involves long sentences containing more explanation on medical issues. Although its topic is on dermatology, TH-D text contains a large number of disease names and medicine names which could be quite helpful in class parameter estimation. About 7.7% words in the vocabulary of this text set are medical terms. Unfortunately, the dataset is also very small with only 37.9k words and 2.6k vocabulary. To further improve medical term coverage, we acquired a dataset of written style hospital reports (MR) containing 1.5 million words and with a 7.4% of its vocabulary being medical terms.

The following table shows statistics about the ratio of medical term in the vocabularies of each dataset we used for language modeling.

*Table 3* Ratio of medical terms in vocabulary of each dataset

| | Vocabulary size(k) | Medical words in vocabulary(k) | Medical vocabulary ratio(%) |
|---|---|---|---|
| TH | 5.6 | 0.6 | 10.7 |
| TH-D | 2.6 | 0.2 | 7.7 |
| SW | 26.5 | 0.3 | 1.1 |
| BN | 34.6 | 0.3 | 0.9 |
| MR | 14.9 | 1.1 | 7.4 |
| CH | 6.9 | 0.2 | 2.9 |

## 5.3 Latent Semantic Analysis (LSA)

Latent Semantic Indexing is a method for dimensionality reduction that maps co-occurring terms onto the same dimensions of the reduced space [9]. It can also be viewed as a method of word co-occurrence analysis. Here its function of co-occurrence analysis is used to compare correlations between different datasets for language model training.

Documents for analysis are transformed into term-by-document matrix $A_{t\times d}$ . SVD projection is then computed by decomposing the document-by-term matrix into the product of three matrices,

$$A_{t\times d} = T_{t\times n} S_{n\times n} (D_{d\times n})^T \qquad (43)$$

in which *T* is a term by dimension matrix, *S* is a matrix of singular values of the SVD with *n* be the smaller one of *t* (number of terms in vocabulary) and *d* (number of documents) and *D* is a document by dimension matrix. Document correlations are calculated as $CoR = B^T B$ where $B = S_{n \times n} D^T_{d \times n}$ with length-normalized columns.

The correlation matrix is in form of document by document matrix with each value $V_{ij}$ in it representing the similarity between the *ith* and *jth* documents. Thus, the correlation matrix is a symmetric one with the diagonal values being 1.0 which implies complete similarity (as shown in tables 4, 5, and 6).

We used LSA in our project with the purpose basically to identify the datasets that are most similar to our target TH dataset, and such datasets may be potentially treated as in-domain ones and therefore trained as a class n-gram language model, as talked about in section 3.2.

All datasets involved in our language modeling task are treated as documents to form the document-by-term matrix for LSA. Not only unigram terms, but also bigram and trigram word sequences are examined to see the relationship of these datasets to the target TH dataset. Table 4 shows such relationships at unigram level, from which we can see a closer relationship between TH and TH-D which can possibly be viewed upon as an in-domain dataset. Correlation coefficients consistent to the unigram case are also seen at bigram level which is shown in table 5 and trigram levels which is shown in table 6.

*Table 4* LSA of unigrams of all datasets

| | TH | TH-D | SW | BN | CH | MR |
|---|---|---|---|---|---|---|
| TH | 1.00 | 0.47 | 0.20 | 0.04 | 0.10 | 0.07 |
| TH-D | 0.47 | 1.00 | 0.89 | 0.02 | 0.06 | 0.001 |
| SW | 0.20 | 0.89 | 1.00 | 0.04 | 0.13 | 0.01 |
| BN | 0.04 | 0.02 | 0.04 | 1.00 | 0.15 | 0.04 |
| CH | 0.10 | 0.06 | 0.13 | 0.15 | 1.00 | 0.03 |
| MR | 0.07 | 0.001 | 0.01 | 0.04 | 0.03 | 1.00 |

*Table 5* LSA of bigrams of all datasets

| | TH | TH-D | SW | BN | CH | MR |
|---|---|---|---|---|---|---|
| TH | 1.00 | 0.68 | 0.07 | 0.08 | 0.07 | 0.01 |
| TH-D | 0.68 | 1.00 | 0. 06 | 0.49 | 0. 11 | 0.001 |
| SW | 0.07 | 0.06 | 1.00 | 0.07 | 0.14 | 0.01 |
| BN | 0.08 | 0.49 | 0.07 | 1.00 | 0.26 | 0.02 |
| CH | 0.07 | 0.11 | 0.14 | 0.26 | 1.00 | 0.02 |
| MR | 0.01 | 0.001 | 0.01 | 0.002 | 0.002 | 1.00 |

*Table 6* LSA of trigrams of all datasets

| | TH | TH-D | SW | BN | CH | MR |
|---|---|---|---|---|---|---|
| TH | 1.00 | 0.25 | 0.06 | 0.05 | 0.03 | 0.01 |
| TH-D | 0.25 | 1.00 | 0.04 | 0.22 | 0.06 | 0.0004 |
| SW | 0.06 | 0.04 | 1.00 | 0.1 | 0.13 | 0.01 |
| BN | 0.05 | 0.22 | 0.1 | 1.00 | 0.2 | 0.002 |
| CH | 0.03 | 0.03 | 0.13 | 0.2 | 1.00 | 0.001 |
| MR | 0.01 | 0.0004 | 0.01 | 0.002 | 0.001 | 1.00 |

# Chapter 6

# Experiments

A series of experiments were conducted in order to compare modeling techniques and corresponding word class definitions. Table 7 is an overview of the experiments performed.

*Table 7* An overview of the experiments performed

| **Experiment target** | **Methods compared** | | |
|---|---|---|---|
| Weight adjustment methods | FWA | EM | |
| LM Smoothing algorithms | Katz | Kneser-Ney | |
| LMs | POS hierarchical back-off LM | Word LM | |
| | Word LM | Class LM | In-domain class out-of-domain word LM |
| Clustering algorithms | Medical term definition method | Automatic clustering algorithm | POS tagging |
| | Medical term definition method | Automatic clustering algorithm on medical terms only | POS tagging on medical terms only |
| In-domain dataset definitions | TH only | TH and TH-D | TH, TH-D and MR |

## 6.1 Datasets description

Since telehealth text dataset drawn from transcripts is too small both in vocabulary size (5.6k) and total word number (119.7k) for training a good language model, we use 5 other datasets coming from other text resources as supplement data, which are telehealth-Dermatology (TH-D), Switchboard (SW), Broadcasting News (BN), Callhome (CH) and Medical Report (MR). The general characteristics about topic and speaking style of each dataset has been described in chapter 5, here we further summarize the properties of each dataset in the following table 8:

*Table 8* A summary of speaking style and topic of each dataset used for language model training

| | Speaking style | Topic |
|---|---|---|
| TH | Conversational unplanned dialogue | medical, about neurophychology |
| TH-D | Conversational unplanned dialogue | medical, about dermatology |
| SW | spontaneous telephone conversational speech | various |
| BN | spontaneous conversational speech from talk shows; read speech in the form of news and voice-overs | various |
| CH | spontaneous telephone conversational speech | various |
| MR | written style, report | medical |

In addition, the number of words contained in each dataset and the size of vocabulary of each of them are shown in Table 9.

*Table 9* Size of each dataset and their vocabulary

| | Total words(k) | Vocabulary size(k) |
|---|---|---|
| TH | 119.7 | 5.6 |
| TH-D | 37.9 | 2.6 |
| SW | 2930.6 | 26.5 |
| BN | 2957.8 | 34.6 |
| MR | 1467.7 | 14.9 |
| CH | 164.8 | 6.9 |
| Total for Training | 7678.5 | 45.4 |

For pure word trigram and pure class trigram language modeling, each dataset is individually used to train language model, and then interpolated to be one large language model for project usage.

For the word-class mixture language modeling, datasets are treated differently based on its property as in-domain or our-of-domain data.

TH and TH-D data, both containing large ratio of medical terms and coming directly from telehealth transcripts, are used for class trigram modeling as in-domain datasets. Among other datasets, SW and BN are two large datasets both containing 2.9M words for training, which are aimed to cover as many word n-gram events as possible. CH, which is a dataset very similar to SW, is used for word trigram language model training also. MR which is a written style text on a medical topic is a dataset that needs to be considered carefully. Containing large amount of medical terms, it is very helpful to cover possible occurring medical terms in practice. Thus, we might be able to treat it as in-domain data, with its medical domain property. However, since it has a totally different style from the target telehealth data, experiments need be performed to see whether it is better to use it as in-domain data for training class trigram language model, or to use it as out-of-domain data for training word trigram language model.

Experiments were also conducted to see how a language model using only telehealth data as in-domain data and all others as out-of-domain data will perform.

## 6.2 Testset description

To test the language models, two test sets are prepared, each of which is taken from a certain doctor's speech and transcripts, we call them Testset1 and Testset2 in this thesis. Testset1 contains 5K words and Testset2 is 2K in size.

## 6.3 Weight Adjustment

Because we need to combine or interpolate language models trained separately from each dataset and to finally form one large mixture language model, how to measure or adjust the weight for each language model in the mixture model using interpolating technique is the problem that needs to be discussed. Usually, the weight is determined by the importance of the language model, or the function it might perform for a certain task. The more relevant a dataset is to the target language, the higher weight the language model got from it will be discussed. There are a couple of ways helpful for determining the weights, which are all tried in our experiments.

### 6.3.1 Forward weight adjustment (FWA)

This is a method that manually adjusts the weights using cross validation set perplexity as a criterion. It borrows the idea of the forward searching method for best feature searching, thus we name it forward weight adjustment. For one test set, the test set perplexity of each individual model is calculated. Language models are ranked according to the perplexities value on the test set in an order from low to high. The top two ranked models with the first and second lowest perplexities are chosen as candidates, and interpolated. Different weights are tried and the best weights are then decided for the two candidates by minimizing test set perplexity value. Then the third ranked model is added and interpolated in the same way, and so for the fourth model and others. This method although does not guarantee to find the optimal weights combinations, however, it is simple and experiments show that it is a fairly practical one, and performs very well.

### 6.3.2 EM algorithm (EM)

EM algorithm is an iterative optimization method to estimate some unknown parameters given measurement data. It contains two steps, E step and M step. The E step is the expectation step which calculates the auxiliary function, $Q = E[P(X, S \mid \Theta)]$ where *X* is observable data, *S* is hidden states and $\Theta$ is the parameter set. The M step, which is maximization step, maximizes *Q* function over previous iteration parameter $\Theta^{t-1}$ to obtain $\Theta$. This method is used in our experiments supposed to find a set of weight values for language models that need to be interpolated by minimizing perplexity.

The E step in our task is to compute the expectation:

$$
\begin{aligned}
Q &= E[\log(P(w_1^N, S \mid \mathrm{A}^t)) \mid w_1^N, A^{t-1}] \\
&= \sum_{s=1}^{k} (P(s \mid w_1^N; \alpha^{t-1}) \log(P(w_1^N \mid s; \alpha^t) \alpha_s)) \qquad (44) \\
&= \sum_{s=1}^{k} P(s \mid w_1^N; \alpha^{t-1}) (\log(P(w_1^N \mid s; \alpha^t) + \log \alpha_s)
\end{aligned}
$$

in which, $w_1^N$ is the word sequence observation, *S* is the hidden state which represents the language model in this case and *k* is the number of language models need to be interpolated. $\alpha^t$ is the weight estimate of current iteration *t*.

To meet the constraint that

$$\sum_{s=1}^{k} \alpha_s = 1 \qquad (45)$$

The function is reformed to:

$$\tilde{Q} = Q + \lambda(\sum_{s=1}^{k} \alpha_s^t - 1) \qquad (46)$$

Then based on (44), (46) is simplified to:

$$\tilde{Q} = \sum_{s=1}^{k} (\log \alpha_s^t) P(s \mid w_1^N ; \alpha^{t-1}) + \lambda(\sum_{s=1}^{k} \alpha_s^t - 1) \qquad (47)$$

To maximize (47), we need to calculate:

$$\frac{\partial \tilde{Q}}{\partial \alpha_s} = \frac{1}{\alpha_s^t} P(s \mid w_1^N ; \alpha^{t-1}) + \lambda = 0 \qquad (48)$$

Solving this equation, we got:

$$\alpha_s^t = \frac{\alpha_s^{t-1} P(w_1^N \mid s)}{\sum_{s=1}^{k} (\alpha_s^{t-1} P(w_1^N \mid s))} \qquad (49)$$

The iteration will stop when convergence is reached, that is, when $\alpha_s^t - \alpha_s^{t-1} < \delta$, where $\delta$ is a threshold. In the experiments, the threshold is set to be 0.01, which is a reasonable small number for convergence.

### 6.3.3 Cross-validation test

Cross validation is a widely used model evaluation method. The main idea is to remove some data from the whole dataset, and use the remaining data for model training. After training process is done, the removed data is then used as evaluation data for checking the performance of the model.

K-fold cross validation is one kind of cross validation tests. The dataset is divided into *k* subsets. Each time, one out of the *k* subsets will be used as the evaluation set, and the other *k-1* sets will be put together to work as the training data. This kind of training and evaluation process will be rerun *k* times, and then the average error across all *k* trials is computed, as the final evaluation result.

In our experiments, as we mentioned in 6.2, we kept a portion of data as test set from each doctor’s dataset, and using the rest data as training data. For the weight adjusting process, we used 10-fold cross validation on the training data to run forward weight adjustment and EM algorithm, and get average weight values for mixture language model training. The reserved test set is later used to test the perplexity and word error rate after the LM is gained by interpolating the models trained from all datasets.

### 6.3.4 Interpolation Weight for Each Dataset

10-fold cross validations were run on TH-D, SW, BN, CH and MR datasets as well as the two sub datasets of TH. Each of the TH dataset is one of the two doctors' speech transcripts notified as Dr.  A and Dr. B. For one run on a certain doctor's data, one of the 10 subsets is held as the evaluation data, while the other 9 subsets are used for training. A language model gained from this training set is then interpolated with other language models got from SW, BN datasets etc. Weights are adjusted using the forward weight adjustment method and EM algorithm on the holdout data. Average weight values are calculated after 10 rounds of this kind of evaluations on each doctor's dataset. The following tables show the results of such adjustment process.

*Table 10* Average weights got using forward weight adjustment by a 10-fold cross validation

| Datasets | Average Weights for each LM gained from corresponding dataset: | | | | | |
|---|---|---|---|---|---|---|
| | TH | TH-D | SW | BN | CH | MR |
| Dr. A | 0.46 | 0.08 | 0.22 | 0.12 | 0.04 | 0.07 |
| Dr. B | 0.45 | 0.06 | 0.24 | 0.1 | 0.06 | 0.09 |

*Table 11* Average weights got from EM algorithm by a 10-fold cross validation

| Datasets | Average weights for each LM gained from corresponding dataset: | | | | | |
|---|---|---|---|---|---|---|
| | TH | TH-D | SW | BN | CH | MR |
| Dr. A | 0.53 | 0.06 | 0.21 | 0.11 | 0.03 | 0.05 |
| Dr. B | 0.61 | 0.04 | 0.19 | 0.07 | 0.04 | 0.05 |

The two sets of weights adjusted using the two methods differ from each other to some extent. The most obvious difference is on the weights given to language model trained from CH dataset. EM algorithm gives more weights to this language model than forward weight adjustment does. To see which method works better in this specific task, trigram word language models interpolated using weights decided by each method are tested in term of test set perplexity. The test set reserved before validation test is now used as the test set. The following table is a comparison between these two weight adjusting methods.

*Table 12* Validation sets PPL on word trigram language models comparing FWA and EM weight adjusting methods

| | testset1 | Testset2 |
|---|---|---|
| FWA | 130.44 | 89.63 |
| EM | 131.56 | 92.47 |

Experiments show that the word trigram language model with interpolation weights decided by EM algorithm got slightly higher perplexity than FWA. Word error rate tests were also done with the language models. The results show no obvious superiority of either LM gained using the FWA method or that gained using EM method. In our following experiments, we interpolated language models using weights adjusted by the FWA.

## 6.4 Experiments on Katz and Kneser-Ney smoothing methods

To see which backoff smoothing is more suitable for language modeling of telehealth conversations, we tried both Katz and Kneser-Hey smoothing algorithms in our experiments. The following table shows comparisons between performances of word trigram language models using Kats smoothing and Kneser-Ney smoothing respectively.

*Table 13* Testset PPL on both word trigram language models

| | testset1 | testset2 |
|---|---|---|
| Katz | 130.58 | 86.38 |
| Kneser-Ney | 130.52 | 89.64 |

From the results shown above, we see that for testset1, Kneser-Ney smoothing method worked a little bit better than Katz smoothing in term of test set perplexity, however, for testset2, Katz smoothing won the competition by 2.6% (absolute). WER rates were tested on the word trigram language models. The results show that, for testset1, Kneser-Ney smoothing method got a lower WER by 1.8% (absolute), while for testset2, Katz smoothing performed better than Kneser-Ney smoothing method by 1.9% (absolute). Thus in model training for telehealth project, we used Kneser-Ney smoothing on language modeling for testset1, and Katz smoothing on language modeling for testset2. However, in subsequent experiments, we used Kneser-Ney smoothing methods for both test sets.

## 6.5 Experiments on POS hierarchical back-off language model

These experiments were conducted on training data from TH, SW, BN, CH and MR datasets only. Three levels of POS hierarchical structure were used in language modeling process, as mentioned in section 4.2. To see how well this kind of language model will work on our telehealth task, we compared it with a word language model also trained from the same datasets. The following table shows the results in test set perplexity and recognition accuracy.

*Table 14* Experiments on test set perplexity of POS Hierarchical backoff language model compared with word language model

| | Testset 1 | Testset 2 |
|---|---|---|
| Word LM | 165.80 | 98.73 |
| POS Hierarchical backoff LM | 220.82 | 179.23 |

The results on test set perplexity show that the POS hierarchical backoff language model didn’t work as well as the word language model trained from the same training set on telehealth task. The possible problem for POS hierarchical backoff language model not working well on the specific task may firstly lie on the difference among the variety of dataset styles. As we discussed in chapter 5, the datasets we used come from text sources of different styles and topics. Texts of all datasets we used are tagged by the same system, while the class n-grams may vary from dataset to dataset. Like in a conversational text set, the following sequence of words and tags might happen

You_PRP are_VBP how_WRB old_JJ

with a much higher frequency than in a written style where it is nearly impossible to see such a sequence because of its grammatical illegality. Such variety of class level n-gram in different datasets may affect the language model performance. Secondly, the POS tagging method we used to give tags to a text corpus based on statistic information was trained from a manually tagged training corpus – Wall Street Journal. The word classes defined in this corpus is mainly based on syntactical rules. However, for text corpora coming from spontaneous conversational speech, such tagging system may tag words incorrectly. Like we showed in previous sections, a phrase "*I am doing good*" which is not grammatically correct however widely used in daily conversation will be tagged as

I_PRP am_VBC doing_VBG good_RB

While "*good*" is a JJ (adjective). The tagging accuracy reported by the mxpostSoftware authors are 96.5% on Wall Street Journal corpus. A small experiment we conducted to see the accuracy of this tagging system on our datasets shows a tagging accuracy about 92%. Incorrect tagging may be a potential problem for this kind of language model too. Lastly, the number of classes we defined using POS tagging is small, with 56 classes in first level, 101 classes in level 2 and only 5 classes in level 3. Over smoothing data into small number of classes may be one of the reasons for the POS language model to lose the competition with word language model, too.

## 6.6 Comparison among clustering algorithms

To see how various clustering algorithms will work on language modeling for this task, we trained class-based language models using several class sets defined by different word clustering methods mentioned in chapter 4. Table 15 shows the test set perplexities resulting from these models. CMed is the class language model based on medical term class definition of Table 2. For class definitions got from automatic clustering and POS tagging techniques, 1000 top frequent words were assigned to singleton classes, that is, a word with the highest number of occurrence in the training data is assigned to a class with only one class member which is the word itself, and the name of the class is presented by the word itself too. C43, C273 and C958 models used automatic clustering method defining 43, 273 and 958 classes, respectively. Adding the 1000 singleton class, they have 1043, 1273 and 1958 word classes for training respectively. POS language model used POS tagging technique defining 120 classes, with the 1000 additional singleton classes, the total number of tags used in training is 1120.

All these class language models were obtained through linear interpolation on language models trained from each dataset. Class language models trained from automatic clustering and combined POS tagging methods failed to beat baseline word language model in term of perplexity. A plausible explanation is that these datasets were

*Table 15* Test set perplexities of LMs defined using different class definitions

| LM | # unigrams | Perplexity | |
|---|---|---|---|
| | | Testset1 | Testset2 |
| Word | 45,5k | 130.52 | 89.64 |
| CMed | 44,2k | 127.73 | 87.92 |
| C43 | 1000 + 43 | 189.39 | 122.41 |
| C273 | 1000 + 273 | 164.52 | 109.50 |
| C958 | 1000 + 958 | 150.45 | 102.99 |
| POS | 1000 + 120 | 196.19 | 129.52 |

of different styles and the class clustering grouped together words that were not highly correlated syntactically or semantically, and therefore over smoothed data. However, class language model trained with medical term class definition decreased perplexity for both test sets over the word trigram language model, by 2.18% (relative) for testset1 and 1.9% (relative) for testset2. Correlated improvements were also seen in speech recognition result - word error rate was decreased by 3.4% (relative) for testset1 and 1.6% (relative) for testset2. This set of experiments showed advantage of grouping only medical terms and semantically similar words into classes, which produced meaningful classes for our specific telehealth domain task.

Another set of experiments was conducted to compare the effect of different word class definitions by grouping only medical terms as well as people's names and digits using the clustering technique, POS tagging method and CMed. Besides the special words that are clustered using these methods, all other words were assigned into singleton classes. Three sets of class definitions are defined using automatic clustering methods, which contain 50, 300 and 1000 classes respectively, and referred as CM50, CM300 and CM1000 models in the following table showing the perplexity results. One set of class definition with 120 classes came from combined POS tagging approach, denoted as POS120 model. The results in Table 16 still show superiority of CMed to other models.

*Table 16* Comparison on using different class definitions that only group medical terms, names and digits

| LM | # unigrams | Perplexity | |
|---|---|---|---|
| | | Testset1 | Testset2 |
| CMed | 44,2k | 127.73 | 87.92 |
| CM50 | 42.2k | 137.95 | 94.03 |
| CM300 | 44.3k | 132.77 | 92.22 |
| CM1000 | 44.5k | 132.16 | 91.31 |
| POS120 | 43.9k | 139.16 | 93.62 |

## 6.7 Comparison among language model combinations

In this set of experiments, language models using different interpolation methods are compared, and perplexity results are shown in Table 17.

*Table 17* Test Set Perplexities of word LM, CMed and CIWO

| | Testset1 | Testset2 |
|---|---|---|
| Word | 130.52 | 89.64 |
| CMed | 127.73 | 87.92 |
| CIWO | 114.64 | 86.16 |

In Table 16, CIWO language model was obtained by interpolating class trigram language models trained from TH and TH-D datasets which were in-domain data with word trigram language models trained from out-of-domain data from other datasets, like SW and BN. Significant improvements were made by such mixture models. CIWO model for testset1 reduced the test set perplexity of the pure class language model CMed by 10.25% (relative), and reduced the test set perplexity of word language model by a 12.17% (relative). On testset2, although improvements were not as striking, CIWO still decreased perplexity by 2% (relative) on pure class model and by 3.9% (relative) over word language model. Tests on recognition accuracy showed consistent results as well, with word error rate of testset1 reduced from the case of pure class language model by 3.4% (relative) and that of baseline word language model by 6.6% (relative). For testset2, CIWO reduced word error rate by 0.8% (relative) over pure class language model and by 2.4% (relative) over word language model.

We also tried to train language models by regarding MR as in-domain data, due to its medical topic property. Also, a mixture language model using only telehealth dataset as in-domain data, while using all other datasets as out-of-domain data was trained and tested. The following table shows a comparison among three mixture language models, defining in-domain and out-of-domain data differently.

*Table 18* Test Set Perplexities of CIWO language model with different definition on in-domain and out-of-domain data

| LM | In-domain | Out-of-domain | Testset1 | Testset2 |
|---|---|---|---|---|
| TH_INDOM | TH | TH-D,SW,BN,CH and MR | 114.14 | 86.23 |
| TH-TH-D_INDOM | TH,TH-D | SW,BN,CH and MR | 114.64 | 86.16 |
| TH-TH-D_MR_INDOM | TH,TH-D and MR | SW,BN,CH | 119.05 | 87.22 |

Results show that language models adding MR data as in-domain data were inferior to those using only TH and TH-D as in-domain data. An explanation is that although MR is a topic related data, its style is quite different from that of the target telehealth task. It is interesting to see inconsistent perplexity results between the two test sets on training language model using only TH data as in-domain data. For testset1, this language model had lower perplexity than the one using TH and TH-D both as in-domain data. However, for testset2, the situation is reversed. Tests on word error rate were made to see how the models work in recognition. And the WER result shows that either for testset1 or for testset2, language models using both TH and TH-D data as in-domain data perform better than those using only TH data as in-domain one.  For testset1, language model with TH combining TH-D as in-domain data beats that using only TH as in-domain data by 0.6% (relative). For testset2, WER was lowered by TH combing TH-D language model by 0.2% (relative).

# Chapter 7

# Conclusions

Language modeling is a significant component element for speech or natural language application, including automatic speech recognition. Statistic n-gram language modeling is a powerful technique for modeling a language. How well a language model will work on a task is influenced by the nature of training data. To train a robust language model, we need enough training data that are similar to the task data in term of speaking style or/and topic domain. Our telehealth project, of a spontaneous conversational styled speech in medical domain, is a challenging task facing difficulty of insufficient in-domain data for language model training. To solve such a problem, efforts were made in this work to utilize data from different sources related to target data source either in text style or in topic domain, and to combine them in an intuitive but efficient way. This thesis mainly focuses on these problems and investigated ways to train such a language model for the specific automatic speech recognition task on telehealth.

## 7.1 Solutions for handling insufficient training data

### 7.1.1 Data from different sources

To enlarge the training data, data from different sources are collected in for 1) reasonable medical terms coverage and 2) n-gram word sequence event estimation. These datasets include conversational Switchboard, Broadcast News and Call Home datasets, written styled Medical Report dataset and telehealth-Dermatology dataset from telehealth but from another medical domain. In order to make a good use of the data, datasets were categorized in term of speaking style and topic. Some statistics were also extracted from the datasets too. And by using latent semantic analysis, similarity between datasets and the target telehealth dataset were measured, which is meaningful for further utilization of these datasets in class or word language modeling process.

### 7.1.2 A novel way to combine in-domain class language model and out-of-domain word language model

Besides choosing data from different data sources, we discussed how to efficiently use these data. Approaches to train basic class trigram language models and hierarchical POS back-off language models were experimented and compared with basic word trigram language models. Based on the characteristics of the specific telehealth project, an intuitive but efficient approach was investigated and proposed in this thesis, which is to interpolate class trigram language model from in-domain datasets and word trigram language models trained from out-of-domain datasets. This kind of mixture language model takes advantages of both word language model which is powerful when enough training data exists and class model which is able to predict unseen events better. By comparing this mixture language model with word trigram and class trigram language models, we proved its reasonability and usefulness.

### 7.1.3 A better word class definition on telehealth

For class-based and POS backoff language models, a couple of word clustering methods, including automatic clustering algorithm, POS tagging were examined, and compared in experiments. For the telehealth project, we proposed our own way defining semantically meaningful word classes and grouping only medical terms and words with obvious semantic resemblance. Definition made in this way considered the properties of the medical domain task, and show superiority in language model performance.

## 7.2 Summery of the Results

On our way toward constructing a language model for the task of automatic speech recognition task in telehealth, dozens of experiments were conducted to compare different language modeling approaches, class definitions, and ways to interpolate language models and so on. Although POS hierarchical language models worked well in some small domain-dependent task, to our telehealth task, it did not work well. For methods combining language models trained from different datasets, experiment results show that the way of combining class-based trigram language models trained from in-domain data and word trigram language models trained from out-of-domain data outperformed word trigram and class trigram language models. This novel way has been proved to be reasonable and helpful to our telehealth project, with significant improvement on test set perplexity and word error rate reduction as well. Regarding word clustering approaches, neither automatic clustering technique nor POS tagging method could beat the word class definitions based on semantics, where only words with highly similar semantic meanings are grouped, because the latter is defined based on the domain properties of this specific task.

Experimental results strongly indicate the efficiency of our language modeling approach involving reasonable combination of in-domain and out-of-domain data, word and class trigram language interpolation and meaningful word class definition.

# Reference List

[1] Bulyko I., Ostendorf M. and Stolcke A., “Getting More Mileage from Web Text Sources for Conversational Speech Language Modeling using Class-dependent Mixtures”, *in Proc. HLT, 2003*

[2] Brown P. et al, “Class-based n-gram Models of Natural Language,” *Computational Linguistics, vol. 18, pp. 467-479, 1992.*

[3] Huang X., Acero A. and Hon H. *Spoken Language Procession – A Guide to Theory, Algorithm and System Development*, Prentice Hall PTR, 2001.

[4] Iyer R., Ostendorf M. and Gish H. “Using Out-of-Domain Data to Improve In-Domain Language Models,” *IEEE Signal Processing Letters, Vol. 4, No. 8, August 1997.*

[5] Jelinek F. “Self-organized Language Modeling for Speech Recognition”, *Readings in Speech Recognition, pp. 450-471*, Morgan Kaufmann Publishers Inc, 1990.

[6] Jelinek F. et al, “Decision Tree Parsing Using a Hidden Derivational Model”, *in proc. The Human Language Technology Workshop, pp. 272-277, 1994.*

[7] Katz S. “Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer”, *IEEE Transactions on Acoustics, Speech, and Signmal Procession, Vol. ASSP-35, pp. 400-401, 1987.*

[8] Kneser R. and Ney. H. “Improved Backing-off for M-Gram Language Modeling,” *in Proc. IEEE ICAASP, pp. 181-184, 1995.*

[9] Manning C. and Schütze H. *Foundations of Statistical Natural Language Processing*, Massachusetts Institute of Technology, 1999.

[10] Merialdo B. “Tagging English Text with a Probabilistic Model”, *Computational Linguistics, 20(2):155-172, 1994.*

[11] Placeway P. et al, “The Estimation of Powerful Language Models from Small and Large Corpora,” *in Proc. ICASSP, volume II, pp. 33-36. 1993.*

[12] Ratnaparkhi A., “A Maximum Entropy Model for Part-of-Speech Tagging”, *In Conference on Empirical Methods in Natural Language Processing, pp. 133-142, 1996.*

[13] Samuelsson D. and Reichl W. “A Class-based Language Model for Large-Vocabulary Speech Recognition Extracted from Part-of-speech Statistics,” *in Proc. ICASSP, Vol.1, pp. 537-540, 1999.*

[14] Weng F., Stolcke A. and Sankar A. “Hub4 Language Modeling Using Domain Interpolation and Data Clustering,” *in Proc. of the DARPA Speech Recognition Workshop, Chantilly, VA, 1997.*

[15] Zitouni I., Siohan O. and Kee C-H. “Hierarchical Class N-Gram Language Models: Towards Better Estimation of Unseen Events in Speech Recognition,” *in Proc. of Eurospeech - Interspeech. pp. 237-240, Geneva, 2003.*

## VITA

Xiaojia Zhang was born on July 14th, 1977, Neijiang, Sichuan, People's Republic of China. After four years of study in Sichuan University, she got a BA degree from Foreign Languages College. In the year of 2000, she came to the United States, and started her study in computer science at University of Missouri – Columbia since 2001. From 2003 spring till now, she's studying in a MS program in Computer Science Department, and hopes that she would get the degree in December 2005.