AI Assisted Language Learning

Hassan Hayat

Preface

This paper comes as a result of me trying to find good applications to help me learn Mandarin Chinese. While different applications have different features, each with their pros and cons, I have yet to find an application that can present truly dynamic content or voice input.

Chinese, in particular, has certain peculiarities that render most learning applications near useless. Chinese does not have an alphabet, instead relies on a system of ideograms where each ideogram represents an idea. An ideogram would translate to a word or phrase in English. The main difficulty with ideograms is that they do not encode pronunciation; they only encode meaning. Because of this, the traditional way of learning the pronunciation of words is by using a mapping of tone and syllables to latin characters called pinyin. While pinyin is useful in learning the pronunciation of words, it is unfortunately useless for all else as no one uses pinyin to read or write.

Unfortunately, applications are over-reliant on pinyin for learning Chinese, especially as Chinese is much easier than most other languages if we exclude writing. Chinese has no verb tenses, no duals or plurals, no feminine vs masculine nouns. Chinese grammar and sentence structure is overall simple and intuitive. As such, applications offer instant gratification by hiding or downplaying the ideograms and focusing on pinyin.

The goal of this research is to demonstrate that it is possible to design applications in such a way as to learn to associate ideograms with pronunciation in a fun and effective way.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to everyone who has helped me make this project successful. I would like to further thank the multitude of language learning applications on the app store that have failed to impress me and effectively help me learn Chinese as they have strongly inspired me to improve on them.

I am also very grateful to all of the researchers and journals who have laid the theoretical foundations for this project. I also wish to particularly the members of the Linguistic Data Consortium (LDC) as well as the trustees of the University of Pennsylvania for keeping such an amazing resource alive. I cannot imagine attempting to take on this project without the LDC as a resource.

Last, but not least, I am deeply grateful to my project instructor and advisor, Prof. Ming-Hwa Wang, for his continued support and encouragement.

Abstract

The paper describes the implementation of an application that assists the user in learning Mandarin Chinese. The application makes use of off-the-shelf speech-to-text recognition and text-to-speech technology to provide an automated solution that improves on existing pedagogic methodologies for learning languages.

Introduction

Learning a new language can be a difficult task, but today we have the technology necessary to design tools for effective computer-aided learning. Chinese poses a particular challenge in learning as the written language does not encode pronunciation and the spoken language does not offer clues as to how a word is spelled. While a textual representation of Chinese vocalization called pinyin exists, it is unused in real world instances as real world Chinese is almost never written in pinyin.

Furthermore, most available applications only take choice-based or textual input. As a result, the learner cannot use these applications to personally practice pronunciation. The only way to do so would be accept voice input, which implies speech-to-text technology.

This study demonstrates the usage of Machine Learning and Deep Learning techniques to design better language learning assistants.

Theoretical Basis and Literature Review

In order to create a language learning application that can give real-time feedback on the user's pronunciation, it is important to implement a robust speech-to-text solution. Unfortunately, speech-to-text is not a trivial problem, especially in a language such as Chinese.

Automatic speech recognition (ASR) is an active and relatively recent research area that is focused in fostering better human-human and human-machine communication. In the past, human-machine communication was restricted to non-verbal forms of interaction such as mouse, keyboard, or touchscreen. This has been mostly due to hardware and technical limitations. However, the computational power available today, through multi-core processors, general purpose graphical processing units (GPGPUs), and CPU/GPU clusters, is several orders of magnitude more than that available just a decade ago. As such, very complex and computational demanding models can be trained within a reasonable timeframe, thus, making ASR suddenly a tractable problem to solve and possible to make it useful for layman use.

The typical architecture of an ASR system has four main components: Signal processing and feature extraction, acoustic model (AM), language model (LM), and hypothesis search. This architecture is a particular form of the general Machine Learning process. It is important to first go over the basics of Machine Learning before going over each of the four components.

Machine Learning Process

In Machine Learning, we start with some input in some arbitrary form. We then have to extract features or variables (or columns in a table) from this input somehow. Given the features, we now have a structured form against which we can train one or more models against. Once the models are trained, they are measured against a test or validation data set. From there we select the model that has performed the best with respect to some appropriate metric, such as the Mean-Squared Error. Now, this model is good to go and can be used with real-world data.

Signal Processing and Feature Extraction

As per the Machine Learning process, the first step is to start with the input and extract features from that input. In the case of ASR, the input is an audio signal. This audio signal has to be processed, the speech enhanced by removing noises and channel distortions, converting the signal from time-domain to frequency-domain, and extracting salient feature variables that are suitable for acoustic models.

Acoustic Model (AM)

Given the features extracted from the signal processing component, the acoustic model integrates knowledge about acoustics and phonetics and generates an AM score for the variable-length feature sequence. This means that the AM represents the relationship between an audio signal and the phonemes, or other linguistic units or building blocks, that make up speech. The AM is usually a multi-layer perceptron trained on a large set of manually transcribed audio recordings of speech, such as broadcast news or conversation data. From this, the AM is able to create a statistical representations of the sounds that make up a word.

Language Model (LM)

The Language Model estimates the probability of a hypothesized word sequence, or LM score, by learning the correlation between words from a training corpora. As such, the LM is a probability distribution over a sequence of words. Language Models are typically implemented as n-gram models. An n-gram is a contiguous sequence of n items from a given sequence of text or speech. An n-gram model is thus a type of probabilistic model for predicting the next item in a sequence in the form of a (n-1) order Markov model. This means that predicting the nth item is solely dependent of the n-1 items in the sequence.

Hypothesis Search

The hypothesis search component combines AM and LM scores given the feature vector sequence and the hypothesized word sequence and outputs the word sequence with the highest score as the recognition result.

Challenges in ASR Modeling

In developing an effective ASR model, the key part one must focus on is the AM. The other components are covered with straightforward models and implementations that can easily be used as black boxes. Unfortunately, AMs have to be specialized per language. That said, more complex mixed-language models do exist, but they are out-of-scope of this paper.

It must be noted that a language like Chinese requires an especially fine-tuned AM due to tones. In Chinese, phonemes consist of syllables and tones, unlike English where syllables suffice to make up a phoneme. Concretely, a syllable like "*mai*" can mean either "to buy" or "to sell" depending on the tone used. If vocalized using the 3rd

tone, an elongated interrogative tone, the word means “to buy”. But, if vocalized using the 4th tone, a short abrupt tone, the word means “to sell”. As such, it is clear that tone recognition is imperative for an effective ASR model geared towards recognizing Chinese. Furthermore, these tones are subject to change for a given word given its adjacent words in a sentence. For example, all characters with 3rd tone should change to 2nd tone when preceding a character with 3rd tone.

Gender-specific models have to be considered for speech recognition and, as such, the training data should be ideally duplicated, one version per gender. Silence also must be modeled as different speakers insert varying amounts of silence in between words and phonemes.

Acoustic Model Implementations

Traditionally, AMs are implemented as either Gaussian Mixture Models (GMM) or GMM - Hidden Markov Models (GMM-HMM). These models form the basis for the more modern and effective Context-Dependent-Deep Neural Network-Hidden Markov Models (CD-DNN-HMM).

Gaussian Mixture Models

A Gaussian Mixture Model (GMM) is a probabilistic model that assumes all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. The Gaussian, or Normal distribution, is the traditional bell curve whose probability density function (PDF) is given by:

$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. Where μ := mean, σ := standard deviation.

As such, one can sample from any number of such Gaussians to form a Gaussian Mixture Model.

As long as no sequence information is taken into account, the GMM is an appropriate model for modeling speech. GMM-based classifiers are highly effective for speaker recognition, denoising speech features, and speech recognition. For speaker recognition, for example, the GMM is directly used as a universal background model for the speech feature distribution pooled from all speakers.

Unfortunately, as soon as speech sequence information is taken into account, the GMM is no longer a good model as it contains no sequence information. In order to be able to take into account sequence information, the model must be aware of past states. Markov Models, through the use of Markov Chains, are such models.

Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical Markov Model in which the system being modeled is assumed to be a Markov Process with unobserved, or hidden, states. A Markov Process, or Markov Chain, is model or process in which the next state is uniquely determined by the present state. This relationship between the next and present state is usually referred to as memorylessness as there is no need in keeping track of prior states. That said, the main trick to using Markov Chains is to realize that it is possible to represent the current states as an aggregation of all the past states. This trick enables one to model future states as a function of all past states using a “memoryless” model.

A Markov Chain is useful when we need to compute a probability for a sequence of events that we can observe in the world. Some tasks however cannot be directly

observed, as such, we need to be able to talk about both observed events and unobserved, or "hidden", events, hence the term Hidden Markov Model.

Formally, a HMM is characterized in terms of its basic elements and parameters.

1. Transition probabilities, $\mathbf{A} = [a_{ij}], i, j = 1,2,...,N$, of a homogeneous Markov Chain with a total of N states

$$a_{ij} = P(q_t = j \,|\, q_{t-1} = i), i, j = 1,2,...,N$$

2. Initial Markov Chain State-Occupation Probabilities: $\pi = [\pi_i], i = 1,2,...,N$, where $\pi_i = P(q_1 = i)$
3. Observation probability distribution, $P(o_t \,|\, s^{(i)}), i = 1,2,...,N$. If o_t is discrete, the distribution associated with each state gives the probabilities of symbolic observations $\{v_1, v_2, \ldots, v_k\}$:

$$b_i(k) = P[o_t = v_k \,|\, q_t = i], i = 1,2,...,N$$

The most common and successful PDF used in speech processing for characterizing the continuous observation probability distribution in the HMM is a multivariate mixture Gaussian distribution for vector-valued observation ($o_t \in \mathbf{R}^D$):

$$b_i(o_t) = \sum_{m=1}^{M} \frac{c_{i,m}}{(2\pi)^{D/2} \,|\, \Sigma_{i,m} \,|^{1/2}} exp[-\frac{1}{2}(o_t - \mu_{i,m})^T \Sigma_{i,m}^{-1} (o_t - \mu_{i,m})]$$

When merged with GMMs, HMMs are very useful in speech modeling and recognition. So called GMM-HMMs have been the traditional models used in the field, but nevertheless have notable weaknesses. These models assume conditional independence as well as piecewise stationarity. Unfortunately, speech tends to be even

more dynamic in the temporal dimension. Different variants of GMM-HMMs have been created over the years in order to try to take into account the extra dynamism found in real-world speech. As one can tell from the above formulas, just standards GMM-HMMs can get complex. Adding extra complexity to such equations can make them unwieldy and difficult to understand. As such, we need a model that can learn the extra complexities inherent in speech without having to explicitly model them in ever-increasingly complex formulas.

Deep Neural Network

A Deep Neural Network (DNN) is a network that maps input to output using nodes and hidden layers. Each node in the network is a neuron that uses a non-linear activation function, such as ReLU. This is a form of supervised learning via back propagation which calculated the error contribution of each neuron after a batch of data is processed.

The general idea behind a deep neural network is very straightforward and very simple to implement. The main challenge lies not in the implementation but in the design of the network architecture and in the pre-processing of the data. Neural networks can have arbitrarily many nodes in arbitrarily many layers with arbitrarily many connections.

DNNs have a very strong classification ability but cannot be used in speech recognition as-is. This is because, while the hidden layers are variable, DNNs require fixed-size inputs. Unfortunately, we need to find a way to handle the variable length of speech signals.

DNN-HMM Hybrid Systems

In a hybrid DNN-HMM system, the dynamics of the input signal is modeled with HMMs and the observation probabilities are estimated through DNNs. Each output neuron of the DNN is trained to estimate the posterior probability of continuous density HMMs' state given the acoustic observations. In addition to their inherently discriminative nature, DNN-HMMs have two advantages: training can be performed simply and efficiently using the Viterbi algorithm and decoding can be performed efficiently. The Viterbi Algorithm is a dynamic programming algorithm used for finding the most likely sequence of hidden states that results in a sequence of observed events. This algorithm is often used in the context of Hidden Markov Models.

More recent advancements in the field have demonstrated that by replacing the neural network with a pre-trained network, significant recognition accuracy improvement can be achieved. These more advanced models are usually referred to as Context-Dependent-Deep Neural Network-Hidden Markov Models (CD-DNN-HMM).

Implementing Speech-to-Text

In order to implement an effective Speech-to-Text solution, the literature seems to indicate that DNN-HMM hybrid systems are the way to go. Given such a solution and sufficient training data, it would then be possible construct an application that recognizes Mandarin Chinese words and phrases. Such an application would greatly improve on the current efforts in Chinese learning applications.

Polyglot Cubed

One such effort is Polyglot Cubed, a game designed at the University of Illinois that is geared towards improving the comprehension of various languages with minimal training. The game is played in multiple rooms filled with floating cubes. Each cube is assigned a word and a pictogram representing the word. The computer then speaks a word. The player must match that word to the corresponding cube.

There are two main issues with this game that can be significantly improved. The first issue is that the game is played in three dimensions. Three dimensional interfaces are very hard to work with, especially in an educational setting. By switching to two dimensions, the experience can be greatly improved. Furthermore, the cubes contain language-agnostic pictograms which must be matched with a computer generated vocalized phoneme. A more valuable approach would be to match computer generated Chinese characters with user speech. As such, the user will be matching relevant information, Chinese text to Chinese sounds. Furthermore, since Chinese characters contain very little to no information relevant to the character's pronunciation, the user automatically demonstrates mastery with every correct match.

Pinyin Equivalence

In this paper, we define the concept of Pinyin Equivalence. Pinyin is the phonetic representation of Chinese characters in a given language, in this case, Mandarin. The set of possible Pinyin representations is much smaller than the set of characters, so multiple characters share the same Pinyin representation. Furthermore, many characters have multiple Pinyin representations (usually at most 3). Two characters are said to be Pinyin-equivalent if they share at least one Pinyin representation.

Furthermore, word A is said to be Pinyin-equivalent to word B if they have the same number of characters and every i-th character in A is Pinyin-equivalent to every i-th character in B.

Hypothesis

The hypothesis of the study is that by introducing matching of user voice input to actual Chinese character, the user will, by definition, know how to match the right sound to the right character and that the application developed through this study will enable such mastery.

Methodology

The game contains three sets of exercises. Each set of exercises represents a set of Chinese characters grouped by level of difficulty. Each exercise contains a Chinese target character, or a short piece of Chinese text, and a microphone button. By clicking on the microphone button, the application can accept voice input. This voice input is then processed by the Web Speech API to generate Chinese text. The target text is then considered to be matched if it is Pinyin-equivalent to the voice-generated text. We represent a matched character in green and a failed match in red.

The player can choose to either do a practice run of each set of exercises or do a full run. Practice runs are not scored and the player can attempt to match the character as many times as the player desires. Furthermore, there is a button to hear the audio representation of the character. The full exercise run on the other hand limits the attempts to 3 per character and keeps a score to track the performance of the player.

Implementation

The game is implemented as a pure frontend web application written in HTML, JS, and CSS. The browsers Google Chrome and Mozilla Firefox ship with the Web Speech API, an API that implements text-to-speech and speech-to-text. To facilitate working with HTML, Facebook's ReactJS framework was used.

The code revolves around two main functions: "transcribeVoice" and "say". "transcribeVoice" returns the text recognized from the microphone using speech-to-text technology. "say" takes in a string of Chinese text and produces an audio representation of the text using text-to-speech technology.

Conclusion

This implementation demonstrates that it is possible to create complex educational applications that leverage state-of-the-art speech-to-text and text-to-speech recognition simply using basic web APIs. This implementation also sets a basis upon which more complex applications can be built. It is now possible to leverage the functionality present to create all sorts of games and exercises to help learn Chinese. The next steps would be to implement a variety of games that use voice as input.

Bibliography

Abdel-Hamid, Ossama. *Automatic Speech Recognition Using Deep Neural Networks: New Possibilities*. Diss. York U, 2014.

Chen, Herng-Yow, and Sheng-Wei Li. "Exploring Many-to-One Speech-to-Text Correlation for Web-Based Language Learning." *ACM Transactions on Multimedia Computing, Communications and Applications* 13th ser. 3.3 (2007).

Grace, Lindsay, Martha Castaneda, and Jeannie Ducher. *User Testing of a Language Learning Game for Mandarin Chinese*. Proc. of Conference on Human Factors in Computing Systems, Texas, USA, Austin. ACM Special Interest Group on Computer-Human Interaction, 2012. Print.

Lamel, Lori, Jean-Luc Gauvain, Viet Bac Le, Ilya Oparin, and Sha Meng. *Improved Models for Mandarin Speech-to-text Transcription*. Proc. of 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Czech Republic, Prague. IEEE, 2011. Print.

Li, Sheng-Wei, Hao-Tung Lin, and Herng-Yow Chen. *How Speech/Text Alignment Benefits Web-based Learning*. Proc. of Demonstration Paper in ACM Multimedia Conference. ACM, 2005. 61-70. Print.

Sharma, Neha, and Shipra Sardana. *A Real Time Speech To Text Conversion System Using Bidirectional Kalman Filter In Matlab*. Proc. of Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India. 2016. Print.

Summerville, Adam, Snodgrass, Sam, Guzdial, Matthew, Holmgård, Christoffer, Hoover, Amy, Isaksen, Aaron, Nealen, Andy, Togelius, Julian. *Procedural Content Generation via Machine Learning (PCGML)*. 2017.

You, Zhen-Jia, Chi-Yuh Shen, Chih-Wei Chang, Baw-Jhiune Liu, and Gwo-Dong Chen. *A Robot as a Teaching Assistant in an English Class*. Proc. of Sixth International Conference on Advanced Learning Technologies. 2006. Print.

Yu, Dong, and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer, 2015. Print.

Appendix: Source Code

lib/chinese.js: The main functions that underlie the application. The rest of the application is just UI code.

```
import han from 'han';

// Function to compare if two characters are pinyin-equivalent
export function charactersPinyinEquivalent(char1, char2) {
  // Get list of possible pinyin representations for char1
  const pinyin1 = han.pinyin(char1)[0];
  // Get list of possible pinyin representations for char2
  const pinyin2 = han.pinyin(char2)[0];
  // Two characters are pinyin-equivalent if they share at least one
  // pinyin representation
  return pinyin1.some((x) => pinyin2.indexOf(x) >= 0)
}


// Function to compare if two words are pinyin-equivalent
export function wordPinyinEquivalent(word1, word2) {
  // Two words are pinyin-equivalent if all characters are pinyin-equivalent
  if (word1.length !== word2.length) {
    // If words are not of the same length, they can't be pinyin-equivalent
    return false;
  } else {
    const charlist1 = word1.split('');
    const charlist2 = word2.split('');
    return charlist1.every(
        (char1, index) => charactersPinyinEquivalent(char1, charlist2[index])
    );
  }
}

// Web Speech API
const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
const SpeechGrammarList = window.SpeechGrammarList || window.webkitSpeechGrammarList;
const SpeechRecognitionEvent = window.SpeechRecognitionEvent ||
window.webkitSpeechRecognitionEvent;
const SpeechSynthesis = window.speechSynthesis;


// Function to get chinese voice
function getVoice() {
  const voices = SpeechSynthesis.getVoices();
  const betterVoice = voices.find(
    (voice) => voice.lang === 'zh-CN' && !voice.default
  );
  const fallbackVoice = voices.find(
    (voice) => voice.lang === 'zh-CN' && voice.default
  );
  return (!!betterVoice) ? betterVoice : fallbackVoice;
}


// Function to say words
export function say(word) {
  const voice = getVoice();
  const utterance = new SpeechSynthesisUtterance(word);
  utterance.voice = voice;
  utterance.rate = 0.7;
  return SpeechSynthesis.speak(utterance)
}
```

```
// Promise to transcribe from voice
export function transcribeVoice(language='zh-cn') {
  const recognition = new SpeechRecognition();
  const speechRecognitionList = new SpeechGrammarList();
  recognition.lang = language;
  recognition.interimResults = false;
  recognition.maxAlternatives = 1;
  return new Promise((resolve, reject) => {
    recognition.onresult = (event) => {
      const last = event.results.length - 1;
      const word = event.results[last][0].transcript;
      resolve(word);
    };
    recognition.onspeechend = () => {
      recognition.stop();
    };
    recognition.onnomatch = (event) => {
      reject('Sound not recognized');
    };
    recognition.onerror = (event) => {
      reject(event.error);
    };
    recognition.start();
  });
}


export const topics = [{
  name: 'HSK Level 1',
  lexicon: [
    "爱", "八", "爸爸", "杯子", "北京", "本", "不", "不客气", "菜", "茶", "吃",
    "出租车", "打电话", "大", "的", "点", "电脑", "电视", "电影", "东西", "都",
    "读", "对不起", "多", "多少", "儿子", "二", "饭店", "飞机", "分钟", "高兴", "个",
    "工作", "狗", "汉语", "好", "号", "喝", "和", "很", "后面", "回", "会", "几",
    "家", "叫", "今天", "九", "开", "看", "看见", "块", "来", "老师", "了", "冷",
    "里", "六", "妈妈", "吗", "买", "猫", "没关系", "没有", "米饭", "名字", "明天",
    "哪", "哪儿", "那", "呢", "能", "你", "年", "女儿", "朋友", "漂亮", "苹果", "七",
    "前面", "钱", "请", "去", "热", "人", "认识", "三", "商店", "上", "上午", "少",
    "谁", "什么", "十", "时候", "是", "书", "水", "水果", "睡觉", "说", "四", "岁",
    "他", "她", "太", "天气", "听", "同学", "喂", "我", "我们", "五", "喜欢", "下",
    "下午", "下雨", "先生", "现在", "想", "小", "小姐", "些", "写", "谢谢", "星期",
    "学生", "学习", "学校", "一", "一点儿", "衣服", "医生", "医院", "椅子", "有",
    "月", "再见", "在", "怎么", "怎么样", "这", "中国", "中午", "住", "桌子", "字",
    "昨天", "坐", "做"
  ]
}, {
  name: 'HSK Level 2',
  lexicon: [
    "吧", "白", "百", "帮助", "报纸", "比", "别", "宾馆", "长", "唱歌",
    "出", "穿", "次", "从", "打篮球", "大家", "到", "得", "等", "弟弟", "第一",
    "懂", "对", "对", "房间", "非常", "服务员务", "高", "告诉", "哥哥", "给",
    "公共汽车", "公司", "贵", "过", "还", "孩子", "好吃", "黑", "红",
    "火车站", "机场", "鸡蛋", "件", "教室", "姐姐", "介绍", "进", "近", "就",
    "觉得", "咖啡", "开始", "考试", "可能", "可以", "课", "快", "快乐", "累", "离",
    "两", "零", "路", "旅游", "卖", "忙", "每", "妹妹", "门", "面条", "男",
    "您", "牛奶", "女", "旁边", "跑步", "便宜", "票", "妻子", "起床", "千", "铅笔",
    "晴", "去年", "让", "日", "上班", "身体", "生病", "生日", "时间", "手表",
    "手机", "说话", "送", "虽然", "但是", "它", "踢足球", "题", "跳舞", "外", "完",
    "玩", "晚上", "往", "为什么", "问", "问题", "西瓜", "希望", "洗", "小时", "笑",
    "新", "姓", "休息", "雪", "颜色", "眼睛", "羊肉", "药", "要", "也", "一起",
    "一下", "已经", "意思", "因为", "所以", "阴", "游泳", "右边", "鱼", "远", "运动",
    "再", "早上", "丈夫", "找", "着", "真", "正在", "知", "准备", "走", "最", "左边"
  ]
}, {
  name: 'HSK Level 3',
```

```
  lexicon: [
    "阿姨", "啊", "矮", "爱好", "安静", "把", "班", "搬", "办法", "办公室", "半",
    "帮忙", "包", "北方", "被", "鼻子", "比较", "比赛", "笔记本", "必须", "变化",
    "别人", "冰箱", "不但", "而且而", "菜单", "参加", "草", "层", "差", "超市",
    "衬衫", "成绩", "城市", "迟到", "除了", "船", "春", "词典", "聪明", "打扫",
    "打算", "带", "担心", "蛋糕", "当然", "地", "灯", "地方", "地铁", "地图",
    "电梯", "电子邮件", "东", "冬", "动物", "短", "段", "锻炼", "多么", "饿", "耳朵",
    "发", "发现", "方便", "放", "放心", "分", "附近", "复习", "干净", "感冒",
    "感兴趣", "刚才", "个子", "根据", "跟", "更", "公斤", "公园", "故事", "刮风",
    "关", "关系", "关心", "关于", "国家", "过", "过去", "还是", "害怕",
    "黑板", "后来", "护照", "花", "花", "画", "坏", "欢迎", "还", "环境", "换",
    "黄河", "回答", "会议", "或者", "几乎", "机会", "极", "记得", "季节", "检查",
    "简单", "健康", "讲", "教", "角", "脚", "接", "街道", "节目", "节日",
    "结婚", "结束", "解决", "借", "经常", "经过", "经理", "久", "旧", "句子",
    "决定", "可爱", "渴", "刻", "客人", "空调", "口", "裤子", "筷子", "蓝",
    "老", "离开", "礼物", "历史", "脸", "练习", "辆", "聊天", "了解", "邻居邻",
    "留学", "楼", "绿", "马", "马上", "满意", "帽子", "米", "面包", "明白", "拿",
    "奶奶", "南", "难", "难过", "年级", "年轻", "鸟", "努力", "爬山", "盘子", "胖",
    "皮鞋", "啤酒", "瓶子", "其实其", "其他", "奇怪", "骑", "起飞", "起来", "清楚",
    "请假", "秋", "裙子", "然后", "热情", "认为", "认真", "容易", "如果", "伞",
    "上网", "生气", "声音", "世界", "试", "瘦", "叔叔", "舒服", "树", "数学",
    "刷牙", "双", "水平", "司机", "太阳", "特别", "疼", "提高", "体育", "甜", "条",
    "同事", "同意", "头发", "突然", "图书馆", "腿", "完成", "碗", "万", "忘记",
    "为", "为了", "位", "文化文", "西", "习惯", "洗手间", "洗澡", "夏", "先",
    "相信", "香蕉", "向", "像", "小心", "校长", "新", "新鲜", "信用卡", "行李箱",
    "熊猫", "需要", "选择", "要求", "爷爷", "一般", "一边", "一定", "一共", "一会儿",
    "一样", "一直", "以前", "音乐", "银行", "饮料", "应该", "影响", "用", "游戏",
    "有名", "又", "遇到", "元", "愿意", "月亮", "越", "站", "张", "长", "着急",
    "照顾", "照片", "照相机", "只", "只", "只有", "才", "中间", "中文", "终于",
    "种", "重要", "周末", "主要", "注意", "自己", "自行车", "总是", "嘴", "最后",
    "最近", "作业"
  ]
}];
```