同声传译

笔声同传传译能够智能的帮助企业提升日常办公效率和准确度。笔声同传系统基于语音识别、语义理解、机器翻译等人工智能技术，结合机构、企业办公应用场景，提供中英文互译、中英文翻译等100个国家语言之间的实时和离线互译；支持会议记录编辑成稿、角色分离、历史文件管理、效果优化等功能。笔声同声传译提供公有云接口及私有化部署方案。

一、产品优势

准确率高

笔声同声传译 AI 引擎普通话识别准确率可达97%。机器翻译核心引擎基于目前先进的 Transformer 模型，翻译可接受度超92%。

安全稳定

笔声同声传译的语音识别引擎日均请求量4亿次，日均处理行业语音5万小时。笔声同声传译的翻译引擎日均翻译请求5亿次，方案部署各垂直行业，在复杂应用环境下均有良好的识别与翻译效果。笔声同声传译系统提供私有云部署和公有云API服务。

定制优化

笔声同声传译提供热词优化和可视化的训练配置页面，用户根据其业务场景自定义完成语音识别、机器翻译结果的优化，和行业场景热词定义，有效提升特定词汇的识别与翻译准确率，满足不同行业定制化的语言需求。

二、应用场景

1. 跨语言交流： 机器翻译可以帮助人们在不同语言之间进行交流，例如跨国商务谈判、国际会议等等。
2. 翻译文档： 机器翻译可以自动翻译各种类型的文档，如合同、电子邮件、新闻报道等等。
3. 跨语言搜索： 机器翻译可以帮助人们在不懂其他语言的情况下搜索和阅读不同语言的网页、文献、资料等等。
4. 语音翻译： 同声传译可以在会议、演讲、研讨会等场合实时翻译，使得不同语言的听众都能够理解发言者的内容。
5. **旅游服务： 机器翻译可以帮助游客在外国旅游时交流和理解当地语言，提供方便。**

三、API文档

1、接口说明

- 集成同声传译时，需按照以下要求:

内容	说明

内容	说明
传输方式	http[s] (为提高安全性，强烈推荐https)
请求地址前缀	http[s]: //translate_v2.abcpen.com/v1/translate/*注：服务器IP不固定，为保证您的接口稳定，请勿通过指定IP的方式调用接口，使用域名方式调用*
接口鉴权	签名机制，详情请参照下方[鉴权说明]
字符编码	UTF-8
响应格式	统一采用JSON格式
开发语言	任意，只要可以向笔声云服务发起HTTP请求的均可

2. 鉴权说明

在调用业务接口时，请求方需要对请求进行签名，服务端通过签名来校验请求的合法性。
鉴权根据application_id, application_secret, timestamp 和signature这几个参数做组合计算，其中application_id, application_secret请在笔声开放平台自主申请。鉴权如下

(1). Python示例代码

```
def generate_signature(application_key: str, application_secret: str) -> str:
    timestamp: str = str(int(time.time()))
    message = f"{application_key}{timestamp}"
    signature = hmac.new(application_secret.encode("utf-8"), message.encode("utf-
8"), hashlib.sha256).hexdigest()
    return signature
```

(2) . Java示例代码

```
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.time.Instant;

public class SignatureGenerator {
    public static String generateSignature(String applicationKey, String
applicationSecret) {
        String timestamp = String.valueOf(Instant.now().getEpochSecond());
        String message = applicationKey + timestamp;
        try {
            Mac sha256Hmac = Mac.getInstance("HmacSHA256");
            SecretKeySpec secretKey = new
SecretKeySpec(applicationSecret.getBytes(StandardCharsets.UTF_8), "HmacSHA256");
            sha256Hmac.init(secretKey);
            byte[] hmacDigest =
sha256Hmac.doFinal(message.getBytes(StandardCharsets.UTF_8));
            return bytesToHex(hmacDigest);
        } catch (NoSuchAlgorithmException | InvalidKeyException e) {
            e.printStackTrace();
            return null;
        }
    }

    private static String bytesToHex(byte[] bytes) {
        StringBuilder hexString = new StringBuilder();
        for (byte b : bytes) {
            String hex = Integer.toHexString(0xff & b);
            if (hex.length() == 1) {
                hexString.append('0');
            }
            hexString.append(hex);
        }
        return hexString.toString();
    }
}
```

(3). Kotlin示例代码

```
import java.security.MessageDigest
import javax.crypto.Mac
import javax.crypto.spec.SecretKeySpec
import java.time.Instant

fun generateSignature(applicationKey: String, applicationSecret: String): String {
    val timestamp = Instant.now().epochSecond.toString()
    val message = "$applicationKey$timestamp"
    val hmac = Mac.getInstance("HmacSHA256")
    hmac.init(SecretKeySpec(applicationSecret.toByteArray(Charsets.UTF_8),
"HmacSHA256"))
    val hmacDigest = hmac.doFinal(message.toByteArray(Charsets.UTF_8))
    return bytesToHex(hmacDigest)
}

private fun bytesToHex(bytes: ByteArray): String {
    val hexChars = CharArray(bytes.size * 2)
    for (i in bytes.indices) {
        val v = bytes[i].toInt() and 0xFF
        hexChars[i * 2] = hexArray[v.ushr(4)]
        hexChars[i * 2 + 1] = hexArray[v and 0x0F]
    }
    return String(hexChars)
}

private val hexArray = "0123456789abcdef".toCharArray()
```

(4). nodejs示例代码

```
const crypto = require('crypto');

function generateSignature(applicationKey, applicationSecret) {
  const timestamp = Math.floor(Date.now() / 1000).toString();
  const message = applicationKey + timestamp;
  const hmac = crypto.createHmac('sha256', applicationSecret);
  hmac.update(message);
  const signature = hmac.digest('hex');
  return signature;
}

// 示例使用
const applicationKey = 'your_application_key';
const applicationSecret = 'your_application_secret';
const signature = generateSignature(applicationKey, applicationSecret);
console.log(signature);
```

(5). go示例代码

```
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
    "time"
)

func generateSignature(applicationKey string, applicationSecret string) string {
    timestamp := fmt.Sprintf("%d", time.Now().Unix())
    message := applicationKey + timestamp
    hmacKey := []byte(applicationSecret)
    hmacData := []byte(message)
    hmacSha256 := hmac.New(sha256.New, hmacKey)
    hmacSha256.Write(hmacData)
    signature := hex.EncodeToString(hmacSha256.Sum(nil))
    return signature
}
```

```
func main() {
    applicationKey := "your_application_key"
    applicationSecret := "your_application_secret"
    signature := generateSignature(applicationKey, applicationSecret)
    fmt.Println(signature)
}
```

(6) Rust示例代码

```
use hmac::{Hmac, Mac, NewMac};
use sha2::Sha256;
use std::time::{SystemTime, UNIX_EPOCH};

fn generate_signature(application_key: &str, application_secret: &str) -> String {
    let timestamp = SystemTime::now()
        .duration_since(UNIX_EPOCH)
        .unwrap()
        .as_secs()
        .to_string();
    let message = format!("{}{}", application_key, timestamp);
    let mut hmac =
        Hmac::<Sha256>::new_varkey(application_secret.as_bytes()).expect("HMAC
initialization failed");
    hmac.update(message.as_bytes());
    let signature = hex::encode(hmac.finalize().into_bytes());
    signature
}

fn main() {
    let application_key = "your_application_key";
    let application_secret = "your_application_secret";
    let signature = generate_signature(application_key, application_secret);
    println!("{}", signature);
}
```

(7) vue示例代码

```
<template>
  <div>
    <button @click="generateSignature">Generate Signature</button>
    <p>Timestamp: {{ timestamp }}</p>
    <p>Signature: {{ signature }}</p>
  </div>
</template>

<script>
import crypto from 'crypto';
```

```
export default {
  name: 'SignatureGenerator',
  data() {
    return {
      applicationKey: 'test1',
      applicationSecret: '2258ACC4-199B-4DCB-B6F3-C2485C63E85A',
      timestamp: null,
      signature: null,
    };
  },
  methods: {
    generateSignature() {
      const timestamp = Math.floor(Date.now() / 1000).toString();
      const message = this.applicationKey + timestamp;
      const hmac = crypto.createHmac('sha256', this.applicationSecret);
      hmac.update(message);
      const signature = hmac.digest('hex');
      this.timestamp = timestamp;
      this.signature = signature;
    },
  },
};
</script>
```

2、鉴权访问

根据上述鉴权说明，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数；将这三个参数放入http(s)请求体头部（header），向服务端发起请求。具体参考后面的示例代码。

3、接口访问

（1）翻译单条语句

/v1/translate/sentence

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
sentence	string	是	无	待翻译语句
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

    - 翻译单条语句{sentence}，从源语言{source_lang}到目标语言{target_lang}。

- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本，是客户待翻译的语句
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

（2）翻译多条语句

/v1/translate/sentences

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
sentences	string	是	无	多条待翻译的语句
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

    - 翻译多条语句{sentences}，从源语言{source_lang}到目标语言{target_lang}。

- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

（2）翻译文件

/v1/translate/file

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
document	File	是	无	上传的待识别的文件
source_lang	string	否	""	为空字符串，引擎自动探测；源语言，比如中文
target_lang	string	是	无	目标语言，如德文

  - 翻译文件{document}，从源语言{source_lang}到目标语言{target_lang}。
- 返回参数定义

参数	类型	备注
src	string	客户输入的源语言文本
target	string	引擎翻译出的目标语言文本
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

（2）语言检测

/v1/translate/language_detection

- 常规参数，属于鉴权信息，生成X-App-Key， X-App-Signature和X-Timestamp这三个参数
- 具体参数定义

参数	类型	是否必须	默认值	备注
text	string	是	无	待探测的文本

  - 输入一段文本，识别出这是哪种语言。
- 返回参数定义

参数	类型	备注
result	string	引擎识别出的某种语言

参数	类型	备注
code	string	状态码, 如“0”， 具体参考[错误码]
msg	string	状态码对应字符串，如"success"
translation_time	Int	翻译消耗时长

4、语言编码

（1）、目前支持100种国家语，这里暂时列出部分国家语言编码。

国家（语言）	编码（简称）	备注
中国	zh	
德国	de	
英文（美国，英国）	en	
俄罗斯	ru	
日语	ja	
法语	fr	
意大利	it	
保加利亚	bg	
越南	vi	
乌克兰	uk	
芬兰	fi	
希伯来语言	he	
马来语（Malay）	ms	
荷兰语	nl	
西班牙语	es	

（2）、具体支持如下国家的语言

```
'Dimili', 'Tuvalu', 'newari classique', 'maori', 'Lule Sami', 'basque', 'tigré',
'mandar', 'grec moderne (après 1453)', 'tereno', 'lezghien', 'massaï', 'sicilien',
'Maori', 'gaélique', 'lunda', 'Palauan', 'Vietnamese', 'navaho', 'ga', 'sango',
'marathe', 'Armenian', 'interlingua (langue auxiliaire internationale)', 'Czech',
'selkoupe', 'kutenai', 'quechua', 'Sepedi', 'australiennes, langues', 'Bliss', 'Cook
Islands Maori', 'Guarani', 'Yiddish', 'Classical Nepal Bhasa', 'dogri', 'Iloko',
"slavon d'église", 'Azerbaijani', 'Nyanja', 'Creek', "grec ancien (jusqu'à 1453)",
'népalais', 'mongol', 'gbaya', 'dimli', 'Gondi', "N'Ko", 'Icelandic', 'cheyenne',
'aymara', 'Acoli', 'Kamba', 'Norwegian Bokmål', 'Norwegian Bokmaal', 'norwegian
bokmål', 'norwegian bokmaal', 'Oirat', 'Kongo', 'bachkir', 'koumyk', 'finno-
ougriennes,langues', 'élamite', 'Alemannic', 'Tigrinya', 'kirmanjki', 'Chipewyan',
'Afro-Asiatic languages', 'Ido', 'yapois', 'Selkup', 'hébreu', 'Kanuri', 'haida',
'sioux, langues', 'yao', 'Occitan (post 1500)', 'Efik', 'Provençal, Old (to 1500)',
'Marwari', 'néerlandais', 'Ossetian', 'turc ottoman (1500-1928)', 'occitan (après
1500)', 'Lingala', 'Pali', 'kalmouk', 'Hungarian', 'cornique', 'Scots', 'kachoube',
'Sorbian languages', 'norrois, vieux', 'suédois', 'Ewondo', 'inupiaq', 'Church
Slavic', 'tibétain', 'Austronesian languages', 'Ndebele, North', 'luxembourgeois',
'Zaza', 'alsacien', 'roumain', 'Norwegian', 'Nzima', 'frison septentrional',
'shona', 'asturoléonais', 'blackfoot', 'napolitain', 'mapuche', 'interlingue',
'banda, langues', 'nigéro-kordofaniennes, langues', 'frison oriental', 'Baltic
languages', 'Mandar', 'touva', 'Interlingue', 'Sichuan Yi', 'galicien', 'sarde',
'edo', 'mongo', 'Kosraean', 'Crimean Tatar', 'Asturian', 'kazakh', 'Chichewa',
'Kapampangan', 'Coptic', 'Sinhala', 'Kwanyama', 'Iroquoian languages', 'Quechua',
'Bemba', 'allemand, moyen haut (ca. 1050-1500)', 'Asturleonese', 'French', 'Tigre',
'Galibi Carib', 'Sardinian', 'Grebo', 'zenaga', 'chamorro', 'manipuri', 'kabardien',
'pedi', 'sami du Nord', 'kirghiz', "amérindiennes de l'Amérique centrale, langues",
'Igbo', 'Salishan languages', 'cherokee', 'umbundu', 'Indic languages', 'Creoles and
pidgins', 'Classical Newari', 'Nepal Bhasa', 'Siouan languages', 'turkmène',
'soninké', 'Mandingo', 'Latin', 'créole haïtien', 'Luba-Lulua', 'Swedish', 'serbe',
'bas allemand', 'chinese_simplified', 'chinese_traditional', 'Tamashek', 'Upper
Sorbian', 'Indonesian', 'sotho du Nord', 'iroquoises, langues', 'Edo', 'Cherokee',
'Ekajuk', 'Bislama', 'kwanyama', 'gothique', 'Pashto', 'Sicilian', 'Romance
languages', 'créoles et pidgins basés sur le français', 'Yupik languages', 'Uighur',
'Tsonga', 'lushai', 'anglais', 'mandingue', 'zandé, langues', 'Pahlavi',
'Kirmanjki', 'Akkadian', 'Shan', 'Nilo-Saharan languages', 'Thai', 'Vai', 'Inari
Sami', 'phénicien', 'chipewyan', 'salishennes, langues', 'Greek, Ancient (to 1453)',
'Ancient Greek', 'ancient greek', 'Kimbundu', 'Dargwa', 'Adangme', 'Tlingit',
'serbo-croate', 'Aleut', 'Nynorsk, Norwegian', 'choctaw', 'Kirdki', 'tigrigna',
'karen, langues', 'sino-tibétaines, langues', 'gaélique écossais', 'Maldivian',
'géorgien', 'slovaque', 'grebo', 'italien', 'avestique', 'gayo', 'tai, langues',
'khasi', 'Tuvinian', 'galla', 'Romansh', 'bikol', 'couchitiques, langues', 'Slavic
languages', 'Pohnpeian', 'sidamo', 'langues non codées', 'German, Old High (ca.750-
1050)', 'Mari', 'non applicable', 'Telugu', 'Yakut', 'bamiléké, langues', 'Spanish',
'Sanskrit', 'Urdu', 'tswana', 'Central Khmer', 'arménien', 'amharique', 'Chinook
jargon', 'Adyghe', 'Avaric', 'Achinese', 'tatar de Crimé', 'Niuean', 'Xhosa',
'javanais', 'zazaki', 'Dimli', 'vote', 'castillan', 'Simplified Chinese', 'bable',
'arapaho', 'kannada', 'Belarusian', 'Chuukese', 'Tereno', 'Ladino', 'Khotanese',
'Hawaiian', 'Southern Altai', 'Khasi', 'éwondo', 'Official Aramaic (700-300 BCE)',
'altaïques, langues', 'moksa', 'vieux slave', 'Tai languages', 'tahitien', 'Kikuyu',
'lojban', 'Chinese (traditional)', 'indéterminée', 'Arumanian', 'Wolaytta', 'temne',
'Wolof', 'Iranian languages', 'Ainu', 'islandais', 'thaï', 'Ga', 'Old Church
Slavonic', 'Sasak', 'macédo-roumain', 'inuktitut', 'magahi', 'Geez', 'Sandawe',
'Karelian', 'nias', 'Kabyle', 'osage', 'French, Middle (ca.1400-1600)', 'Arabic',
'sémitiques, langues', "réservée à l'usage local", 'Fang', 'Zhuang', 'Limburgish',
"mi'kmaq", 'Washo', 'bouriate', 'marvari', 'Walloon', 'Not applicable', 'Herero',
'afar', 'muskogee', 'syriaque classique', "Gwich'in", 'xhosa', 'multilingue',
'Caucasian languages', 'Micmac', 'afrihili', 'ijo, langues', 'singhalais',
'samaritain', 'efik', 'créoles et pidgins basés sur le portugais', 'finnois',
'tlhIngan-Hol', 'Hiri Motu', 'slaves, langues', 'Kalaallisut', 'Swahili',
'Limburgan', 'haïtien', 'Tokelau', 'wolaitta', 'pahlavi', 'lamba', 'Welsh', 'palau',
'Dravidian languages', 'allemand, vieux haut (ca. 750-1050)', 'Low Saxon', 'nahuatl,
langues', 'celtes, langues', 'luo (Kenya et Tanzanie)', 'ekajuk', 'Basque', 'avar',
'Western Pahari languages', 'sepedi', 'letton', 'chuang', 'chichewa', 'tadjik',
'kosrae', 'néerlandais moyen (ca. 1050-1350)', 'Kawi', 'fanti', 'ndonga',
'aragonais', 'nordamérindiennes, langues', 'Haitian', 'kikuyu', 'Kara-Kalpak',
'Gothic', 'kashmiri', 'nynorsk, norvégien', 'Malayalam', 'wallon', 'Soninke',
'Macedo-Romanian', 'Interlingua (International Auxiliary Language Association)',
'sranan tongo', 'papoues, langues', 'Divehi', 'norwegian_bokmal', 'Newari',
'Occidental', 'hmong', 'Moldavian', 'vieux bulgare', 'Sign Languages', 'symboles
Bliss', 'Kuanyama', 'sami de Lule', 'judéo-persan', 'Caddo', 'Finnish', 'Judeo-
Persian', 'Zande languages', 'kongo', 'Catalan', 'Chechen', 'Nuosu', 'Skolt Sami',
'Arawak', 'Marshallese', 'Chibcha', 'allemand', 'Tagalog', 'moré', 'Lower Sorbian',
'Neapolitan', 'Serbian', 'dakota', 'papiamento', 'ido', 'kurde', 'Rarotongan',
'Bable', 'Nauru', 'Lithuanian', 'Bini', 'Greenlandic', 'Mapudungun', 'sumérien',
'South American Indian languages', 'bemba', 'Inupiaq', 'German', 'sames, langues',
'Karen languages', 'rwanda', 'Nyoro', 'azéri', 'Tonga (Nyasa)', 'malayalam',
'coréen', 'oriya', 'Kumyk', 'Philippine languages)', 'anglais moyen (1100-1500)',
'tagalog', 'Altaic languages', 'pohnpei', 'Traditional Chinese', 'Tumbuka',
'fidjien', 'tiv', 'Dzongkha', 'Songhai languages', 'Luo (Kenya and Tanzania)',
'Prakrit languages', 'Lao', 'prâkrit, langues', 'iakoute', 'judéo-arabe', 'Nogai',
'Siksika', 'frison occidental', 'Dogri', 'maori des îles Cook', 'tokelau', 'German,
Middle High (ca.1050-1500)', 'tupi, langues', 'birman', 'akkadien', 'Manx',
'Danish', 'breton', 'fang', 'Alsatian', 'Kachin', 'Udmurt', 'herero', 'Sundanese',
'Khoisan languages', 'maya, langues', 'Madurese', 'Sotho, Southern', 'Slovenian',
'goudjrati', 'soussou', 'Eastern Frisian', 'Votic', 'krou, langues', 'Rapanui',
'Castilian', 'Turkmen', 'nilo-sahariennes, langues', 'luba-katanga', 'Akan',
'nogaï', 'estonien', 'Fijian', 'zuni', 'luba-lulua', 'Crimean Turkish', 'Wakashan
languages', 'nyankolé', 'Artificial languages', 'Uzbek', 'Greek', 'Manobo
languages', 'khotanais', 'aroumain', 'Turkish, Ottoman (1500-1928)', 'Hittite',
'newari', 'carib', 'Georgian', 'Aromanian', 'Luiseno', 'carélien', 'Oromo', 'tatar',
'French, Old (842-ca.1400)', 'peul', 'Niger-Kordofanian languages', 'léonais',
'Chamic languages', 'frioulan', 'mapudungun', 'russe', 'Tahitian', 'bichlamar',
'Dyula', 'Munda languages', 'indonésien', 'wakashanes, langues', 'Uncoded
languages', 'kachin', 'samoan', 'norvégien bokmål', 'sandawe', 'South Ndebele', 'Ijo
languages', 'Australian languages', 'Blin', 'abkhaze', 'flamand', 'judéo-espagnol',
'madourais', 'esclave (athapascan)', 'Portuguese', 'jingpho', 'Luba-Katanga',
'Luxembourgish', 'Somali', 'kanouri', 'washo', 'baltes, langues', 'Breton', 'Old
Newari', 'Gujarati', 'langues himachalis', 'galibi', 'igbo', 'guèze', 'Bashkir',
'Volapük', 'mounda, langues', 'Zazaki', 'sotho du Sud', 'venda', 'Oriya', 'Scottish
Gaelic', 'tlingit', 'Delaware', 'germaniques, langues', 'Hiligaynon', 'Moksha',
'Baluchi', 'No linguistic content', 'Mirandese', 'Bikol', 'Tonga (Tonga Islands)',
'Maithili', 'tamoul', 'copte', 'pas de contenu linguistique', 'Haida', 'français
ancien (842-ca.1400)', 'Bhojpuri', 'Bantu languages', 'mannois', 'arabe', 'Northern
Sotho', 'Venda', 'Erzya', 'dargwa', 'Jingpho', 'makassar', 'romanes, langues',
'Sango', 'tok pisin', 'irlandais', 'Sinhalese', 'Tatar', 'français', 'Albanian',
"occitan ancien (jusqu'à 1500)", 'ouïgour', 'Sami languages', 'Umbundu', 'Afar',
'Chamorro', 'batak, langues', 'sogdien', 'Tiv', 'Iban', 'yupik, langues',
'kimbundu', 'vietnamien', 'kuanyama', 'Norwegian Nynorsk', 'norwegian nynorsk',
'zaza', 'Sindhi', 'Pedi', 'Estonian', 'Ojibwa', 'maithili', 'philippines, langues',
'Komi', 'Undetermined', 'chibcha', 'Mongolian', 'Ewe', 'Land Dayak languages', 'sami
du Sud', 'Magahi', 'Ossetic', 'ouszbek', 'bas-sorabe', 'Serbo-Croatian', 'Kpelle',
'Afrikaans', 'nyanja', 'Tsimshian', 'Filipino', "provençal ancien (jusqu'à 1500)",
'mari', 'chewa', 'sérère', 'Nepali', 'télougou', 'chuuk', 'Tok Pisin', 'tchétchène',
'Hmong', 'Sakan', 'Ingush', 'ganda', 'yi de Sichuan', 'khmer central', 'japonais',
'Kannada', 'Sidamo', 'lahnda', 'Yapese', 'Apache languages', 'Hupa', 'Rundi',
'Brasilian Portuguese', 'baloutchi', 'niué', 'Masai', 'saxon, bas', 'mohawk',
'Mohawk', 'Gikuyu', 'Persian', 'Old Bulgarian', 'Kru languages', 'Judeo-Arabic',
'caddo', 'Fulah', 'ilocano', 'pendjabi', 'Dene Suline', 'Dogrib', 'Zulu', 'Ndebele,
South', 'klingon', 'Papiamento', 'Russian', 'celtiques, langues', 'Samoan',
'Awadhi', 'Mong', "Mi'kmaq", 'turc', 'Hindi', 'Esperanto', 'persan', 'brasilian
portuguese', 'irlandais moyen (900-1200)', 'Mon-Khmer languages', 'Dakota',
'Javanese', 'konkani', 'soundanais', 'tsimshian', 'tchèque', 'Pangasinan', 'bedja',
'Dutch, Middle (ca.1050-1350)', 'ougaritique', 'khoïsan, langues', 'twi', 'dayak,
langues', 'Western Frisian', 'unknown', 'Aymara', 'écossais', 'Gbaya',
'Blissymbolics', 'Lozi', 'hindi', 'albanais', 'Chagatai', 'yiddish', 'Nias',
'Bambara', 'dravidiennes,langues', 'allemand, bas', 'djaghataï', 'Northern Frisian',
'Yao', 'portugais', 'Reserved for local use', 'lingala', 'kirdki', 'altai du Sud',
'Dhivehi', 'fon', 'Berber languages)', 'Croatian', 'Tajik', 'luiseno', 'sasak',
'hongrois', 'Manchu', 'bugi', 'Faroese', 'pampangan', 'Kurukh', 'ndébélé du Sud',
'angika', 'syriaque', 'Buriat', 'Southern Sami', 'Dinka', 'Moldovan', 'Hebrew',
'ndébélé du Nord', 'Semitic languages', 'pangasinan', 'sukuma', 'tumbuka', 'aïnou',
'ukrainien', 'indo-aryennes, langues', 'Avestan', 'français moyen (1400-1600)',
'Klingon', 'Gorontalo', 'nogay', 'Classical Syriac', 'Lushai', 'Cushitic languages',
'Minangkabau', 'awadhi', 'Flemish', 'Inuktitut', 'Algonquian languages', 'kurukh',
'Tamil', 'Tswana', 'bosniaque', 'balinais', 'Cheyenne', 'Mende', 'sakan', 'ourdou',
'hiligaynon', 'Kurdish', 'perse, vieux (ca. 600-400 av. J.-C.)', 'Tupi languages',
'Pilipino', 'valencien', 'Banda languages', 'erza', 'micmac', 'chinois', 'afro-
asiatiques, langues', 'Lunda', 'Kyrgyz', 'Malay', 'Gayo', 'môn-khmer, langues',
'mirandais', 'algonquines, langues', 'Ugaritic', 'Romanian', 'Irish', 'Nahuatl
languages', 'Nubian languages', 'Manipuri', 'Burmese', 'bilen', 'bengali',
'Kinyarwanda', 'Central American Indian languages', 'bhojpuri', 'Kutenai', 'Cree',
'kom', 'Kirghiz', 'Gaelic', 'iban', 'Mapuche', 'kpellé', 'Batak languages',
'austronésiennes, langues', 'haut-sorabe', 'zoulou', 'Letzeburgesch', 'gond',
'Creoles and pidgins, French-based', 'Amharic', 'delaware', 'Ganda', 'bas saxon',
'Sranan Tongo', 'tsonga', 'tchouvache', 'bulgare', 'Swiss German', 'Leonese',
'Japanese', 'Corsican', 'gorontalo', 'Persian, Old (ca.600-400 B.C.)', 'Serer',
'malais', 'karib', 'chames, langues', 'Saxon, Low', 'danois', 'hittite', 'sud-
amérindiennes, langues', 'guarani', 'Chuvash', 'wolaytta', 'swahili', 'Timne',
'rundi', 'Galician', 'Irish, Old (to 900)', 'lozi', 'Indo-European languages',
'espagnol', 'volapük', "créoles et pidgins basés sur l'anglais", 'Punjabi', 'nepal
bhasa', 'Germanic languages', 'tsigane', 'sanskrit', 'Syriac', 'Wolaitta',
'ingouche', 'Celtic languages', 'Afrihili', 'Lamba', 'kamba', 'asturien',
'Phoenician', 'cree', 'chinook, jargon', 'Polish', 'nauruan', 'dioula', 'Zapotec',
'songhai, langues', 'Waray', 'cebuano', 'Bulgarian', 'dimili', 'Imperial Aramaic
(700-300 BCE)', 'créoles et pidgins', 'langues des signes', 'Angika', 'pali',
'waray', 'Friulian', 'alémanique', 'Cornish', 'Norse, Old', 'Buginese', 'Valencian',
"araméen d'empire (700-300 BCE)", 'zapotèque', 'Northern Sami', 'Ukrainian',
'douala', 'Mongo', 'Uyghur', 'slovène', 'mandchou', 'Slovak', 'lao', 'Fanti',
'Sumerian', 'tongan (Îles Tonga)', 'Occitan, Old (to 1500)', 'Gilbertese', 'Fon',
'malgache', 'Navajo', 'maltais', 'adyghé', 'moldave', 'braj', 'North American Indian
languages', 'artificielles, langues', 'dzongkha', 'akan', 'suisse alémanique',
'afrikaans', 'Arapaho', 'Chinese', 'Beja', 'Pampanga', 'Samaritan Aramaic', 'Creoles
and pidgins, Portuguese-based', 'nubiennes, langues', 'Twi', 'Bilin', 'Bengali',
'marshall', 'apaches, langues', 'macédonien', 'iraniennes, langues', 'aléoute',
'nyamwezi', 'Zuni', 'slavon liturgique', 'kiribati', 'swati', 'Adygei', 'Latvian',
'athapascanes, langues', 'manobo, langues', 'tetum', 'mapuce', 'ojibwa', 'éwé',
'Osage', 'latin', 'Sogdian', 'Egyptian (Ancient)', 'Duala', 'catalan', 'Cebuano',
"irlandais ancien (jusqu'à 900)", 'Bedawiyet', 'minangkabau', 'Pushto', 'croate',
'rajasthani', 'gallois', 'Chewa', 'Shona', 'maldivien', 'Abkhazian', 'wolof',
'karakalpak', 'polonais', 'Santali', 'ossète', 'Church Slavonic', 'Slave
(Athapascan)', 'bini', 'tonga (Nyasa)', 'Kalmyk', 'Basa', 'limbourgeois',
'Assamese', 'Italian', 'tuvalu', 'sindhi', 'Irish, Middle (900-1200)', 'Sukuma',
'filipino', 'égyptien', 'langues paharis occidentales', 'sorabes, langues',
'English', 'Mayan languages', 'Malagasy', 'Yoruba', 'Slovene', 'oudmourte',
'adangme', 'Turkish', 'Panjabi', 'berbères, langues', 'Finno-Ugrian languages)',
'haoussa', 'bambara', 'kawi', 'German, Low', 'Rajasthani', 'hawaïen', 'acoli',
'assamais', 'somali', 'Zenaga', 'Mossi', 'Kabardian', 'hupa', 'Greek, Modern
(1453-)', 'Kashmiri', 'norvégien nynorsk', 'rarotonga', 'Blissymbols', 'Bokmål,
Norwegian', 'Nyamwezi', 'Susu', 'Elamite', 'lituanien', 'Kashubian', 'espéranto',
'Sino-Tibetan languages', 'basa', "n'ko", 'Aragonese', 'groenlandais', 'rapanui',
'Navaho', 'pachto', 'biélorusse', 'Bosnian', 'tamacheq', 'Swati', 'sami skolt',
'norvégien', 'yoruba', 'Low German', 'Papuan languages', 'Bamileke languages',
"gwich'in", 'Old Slavonic', "sami d'Inari", 'Limburger', 'Creoles and pidgins,
English based', 'kabyle', 'North Ndebele', 'otomi, langues', 'pilipino', 'manx',
'hiri motu', 'Nyankole', 'Maltese', 'Makasar', 'dogrib', 'karatchai balkar',
'féroïen', 'Macedonian', 'Lezghian', 'Ndonga', 'bantou, langues', 'Konkani',
'Balinese', 'santal', 'Chuang', 'aceh', 'Tetum', 'oïrat', 'Lojban', 'zhuang',
'mendé', 'Multiple languages', 'Romany', 'Bihari languages', 'dinka', 'Himachali
languages', 'Athapascan languages', 'Haitian Creole', 'indo-européennes, langues',
'Korean', 'Braj', 'corse', 'nzema', 'Kazakh', 'romanche', 'Marathi', 'arawak',
'nyoro', 'Dutch', 'blin', 'Tibetan', 'caucasiennes, langues', 'Otomian languages',
'Hausa', 'Karachay-Balkar', 'langues biharis', 'Choctaw', 'Lahnda', 'chan', 'vaï'
```

5、接口调用实例

注意，实际使用中，请加上鉴权部分的代码，否则客户端无法验证通s过

（1）、Python实例

```
import http3
import requests
import json
from time import perf_counter
from requests.utils import quote

#URL_SERVER = "http://192.168.2.201:3701"
URL_SERVER = "http://translate_v2.abcpen.com"
```

```
sentences_zh = """"
    首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主客户，ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，
在近年来，国内的半导体企业数量飙升，全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大量出口给中
国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。
    """
multi_sentences = ["Dies ist ein deutscher Satz", "This is an English sentence", "这是一个中文句子"]

def translate_sentence(sentence, source_lang, target_lang):
    values = {'sentence': sentence.encode(
        "utf8"), 'source_lang': source_lang, 'target_lang': target_lang}
    url = f"{URL_SERVER}/v1/translate/sentence"
    t1 = perf_counter()
    r = requests.post(url, data=values)
    print(f"=======================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")


def translate_sentences(sentence, source_lang, target_lang):
    values = {'sentences': sentence.encode(
        "utf8"), 'source_lang': source_lang, 'target_lang': target_lang}
    print(f"Json values: {values}")
    url = f"{URL_SERVER}/v1/translate/sentences"
    t1 = perf_counter()
    r = requests.post(url, data=values)
    print(f"=======================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")

def translate_file(file, source_lang, target_lang):
    files = {'document': open(f'./{file}', 'rb')}
    json = {"target_lang": target_lang}
    url = f"{URL_SERVER}/v1/translate/file"
    t1 = perf_counter()
    r = requests.post(url, files=files, data=json)
    print(f"=======================================> Time consume: {perf_counter()-t1}s, Result: {r.text}")

def translate_detection(sentences):
    values = {'text': sentences}
    print(f"Json values: {values}")
    url = f"{URL_SERVER}/v1/translate/language_detection"
    t1 = perf_counter()
    r = requests.post(url, json=values)
    print(f"=======================================> Language Detection time consume: {perf_counter()-t1}s, Result: {r.text}")

    for text in sentences:
        r = requests.get(f"{URL_SERVER}/v1/translate/language_detection?text="+quote(text))
        print(text, "language detect result==>", r.json())

if __name__ == "__main__":
    t1 = perf_counter()
    for item in range(1):
        translate_sentence(sentences_zh, 'zh', 'en')
        print("\n\n")
        translate_sentences(sentences_zh, 'zh', 'en')
        print("\n\n")
        translate_file('news_1.txt', 'zh', 'en')
        print("\n\n")
        translate_detection(multi_sentences)
        print("\n\n")
    t2 = perf_counter()
    print(f"Total time: {t2-t1}s, average time: {(t2-t1)/100}s")
```

(2) 、Java实例

```
import okhttp3.*;
import org.jetbrains.annotations.NotNull;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.Arrays;
import java.util.List;

public class TranslationClient {
    private static final String URL_SERVER = "http://translate_v2.abcpen.com";
private static final MediaType JSON_MEDIA_TYPE = MediaType.get("application/json;
charset=utf-8");
private static final MediaType BINARY_MEDIA_TYPE = MediaType.get("application/octet-
stream");

private final OkHttpClient httpClient;

public TranslationClient() {
    this.httpClient = new OkHttpClient();
}

public void translateSentence(String sentence, String sourceLang, String targetLang)
{
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/sentence").newBuilder().build();
    String json = String.format("
{\"sentence\":\"%s\",\"source_lang\":\"%s\",\"target_lang\":\"%s\"}",
            sentence, sourceLang, targetLang);
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void translateSentences(String sentences, String sourceLang, String
targetLang) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/sentences").newBuilder().build();
    String json = String.format("
{\"sentences\":\"%s\",\"source_lang\":\"%s\",\"target_lang\":\"%s\"}",
            sentences, sourceLang, targetLang);
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void translateFile(File file, String sourceLang, String targetLang) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/file").newBuilder().build();
    RequestBody body = new MultipartBody.Builder()
            .setType(MultipartBody.FORM)
            .addFormDataPart("target_lang", targetLang)
            .addFormDataPart("document", file.getName(),
                    RequestBody.create(file, BINARY_MEDIA_TYPE))
            .build();
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Time consume: %.3fs, Result: %s%n",
response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
```

```java
}

public void detectLanguage(String... sentences) {
    HttpUrl url = HttpUrl.parse(URL_SERVER +
"/v1/translate/language_detection").newBuilder().build();
    String json = String.format("{\"text\": %s}", toJsonArray(sentences));
    RequestBody body = RequestBody.create(json, JSON_MEDIA_TYPE);
    Request request = new Request.Builder().url(url).post(body).build();
    try (Response response = httpClient.newCall(request).execute()) {
        if (!response.isSuccessful()) {
            throw new IOException("Unexpected code " + response);
        }
        String result = response.body().string();
        System.out.printf("Language Detection time consume: %.3fs, Result: %s%n",
                response.receivedResponseAtMillis() / 1000d, result);
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private String toJsonArray(String[] values) {
    StringBuilder builder = new StringBuilder("[");
    for (String value : values) {
        builder.append('"').append(value).append('"').append(",");
    }
    builder.deleteCharAt(builder.length() - 1);
    builder.append("]");
    return builder.toString();
}

public static void main(String[] args) {
    String sentence = "首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大
批金主客户，" +
            "ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数
量飙升，" +
            "全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然
在之前一直未大量出口给中国光刻机，" +
            "但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。";
    List<String> multiSentences = Arrays.asList(
            "Dies ist ein deutscher Satz",
            "This is an English sentence",
            "这是一个中文句子");

    TranslationClient client = new TranslationClient();

    client.translateSentence(sentence, "zh", "en");
    System.out.println();

    client.translateSentences(sentence, "zh", "en");
    System.out.println();

    File file = new File("./news_1.txt");
    client.translateFile(file, "zh", "en");
    System.out.println();

    client.detectLanguage(multiSentences.toArray(new String[0]));
    System.out.println();
}}
```

(3) 、Nodejs实例

```
const axios = require('axios');
const FormData = require('form-data');
const fs = require('fs');

const URL_SERVER = "http://translate_v2.abcpen.com";

async function translateSentence(sentence, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/sentence;
const data = { sentence: sentence, source_lang: sourceLang, target_lang: targetLang
};
try {
const response = await axios.post(url, data);
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function translateSentences(sentences, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/sentences;
const data = { sentences: sentences, source_lang: sourceLang, target_lang:
targetLang };
try {
const response = await axios.post(url, data);
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function translateFile(filename, sourceLang, targetLang) {
const url = ${URL_SERVER}/v1/translate/file;
const form = new FormData();
form.append('target_lang', targetLang);
form.append('document', fs.createReadStream(filename));
try {
const response = await axios.post(url, form, { headers: form.getHeaders() });
console.log(Time consume: ${response.headers['x-response-time']}s, Result:
${response.data});
} catch (error) {
console.error(error);
}
}

async function detectLanguage(sentences) {
const url = ${URL_SERVER}/v1/translate/language_detection;
const data = { text: sentences };
try {
const response = await axios.post(url, data);
console.log(Language Detection time consume: ${response.headers['x-response-
time']}s, Result: ${response.data});
} catch (error) {
console.error(error);
}
}

async function main() {
const sentence = "首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批金主
客户，" +
"ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数量飙升，" +
"全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前一直未大
量出口给中国光刻机，" +
"但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。";
const multiSentences = ["Dies ist ein deutscher Satz", "This is an English
sentence", "这是一个中文句子"];

await translateSentence(sentence, 'zh', 'en');
console.log();

await translateSentences(sentence, 'zh', 'en');
console.log();

await translateFile('./news_1.txt', 'zh', 'en');
console.log();

await detectLanguage(multiSentences);
console.log();
}

main();
```

(4) 、Kotlin实例

```
import okhttp3.*
import okhttp3.MediaType.Companion.toMediaTypeOrNull
import okhttp3.RequestBody.Companion.asRequestBody
import java.io.File
import java.io.IOException
```

```
class TranslationClient {
private val URL_SERVER = "http://translate_v2.abcpen.com"
private val JSON_MEDIA_TYPE = "application/json".toMediaTypeOrNull()
    fun translateSentence(sentence: String, sourceLang: String, targetLang: String)
{
    val url = "${URL_SERVER}/v1/translate/sentence"
    val json = """{"sentence": "$sentence", "source_lang": "$sourceLang",
"target_lang": "$targetLang"}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun translateSentences(sentences: String, sourceLang: String, targetLang: String) {
    val url = "${URL_SERVER}/v1/translate/sentences"
    val json = """{"sentences": "$sentences", "source_lang": "$sourceLang",
"target_lang": "$targetLang"}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun translateFile(file: File, sourceLang: String, targetLang: String) {
    val url = "${URL_SERVER}/v1/translate/file"
    val body = MultipartBody.Builder().setType(MultipartBody.FORM)
        .addFormDataPart("target_lang", targetLang)
        .addFormDataPart("document", file.name, file.asRequestBody())
        .build()
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println("Time consume: ${response.header("x-response-time")}s, Result:
${response.body?.string()}")
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

fun detectLanguage(sentences: Array<String>) {
    val url = "${URL_SERVER}/v1/translate/language_detection"
    val json = """{"text": ${toJsonArray(sentences)}}"""
    val body = json.toRequestBody(JSON_MEDIA_TYPE)
    val request = Request.Builder().url(url).post(body).build()
    try {
        val response = OkHttpClient().newCall(request).execute()
        if (!response.isSuccessful) throw IOException("Unexpected code $response")
        println(
            "Language Detection time consume: ${response.header("x-response-
time")}s, " +
                    "Result: ${response.body?.string()}"
        )
    } catch (e: IOException) {
        e.printStackTrace()
    }
}

private fun toJsonArray(values: Array<String>): String {
    val builder = StringBuilder("[")
    for (value in values) {
        builder.append("\"").append(value).append("\",")
    }
    builder.deleteChar
    builder.append("]")
    return builder.toString()
}
}
fun main() {
val sentences_zh = """首先，ASML作为全球最大的光刻机制造厂商，尽管能够领跑全世界，可如果没有大批
金主客户，""" +
"""ASML也不会过得那么舒坦。中国市场作为全球最大的消费市场，在近年来，国内的半导体企业数量飙升，"""
+
"""全球每新增20家半导体企业，就有19家是中国的，可见中国市场的巨大潜力。ASML也不傻，虽然在之前""" +
"""一直未大量出口给中国光刻机，但是随着中国对DUV光刻机需求的增长，ASML也开始重视起中国市场了。"""
val multi_sentences = arrayOf("Dies ist ein deutscher Satz", "This is an English
sentence", "这是一个中文句子")
    val client = TranslationClient()
client.translateSentence(sentences_zh, "zh", "en")
println()

client.translateSentences(sentences_zh, "zh", "en")
println()

val file = File("news_1.txt")
client.translateFile(file, "zh", "en")
println()

client.detectLanguage(multi_sentences)
println()
```

```
}
```

6. 错误码

错误码	描述	说明	处理方式
0	success	成功	
-1	in progress	识别中	请继续重试
-2	audio encode error	音频编码错误	请编码成正确的格式，再提交请求
10105	illegal access	没有权限	检查apiKey，ip，ts等授权参数是否正确
10106	invalid parameter	无效参数	上传必要的参数， 检查参数格式以及编码
10107	illegal parameter	非法参数值	检查参数值是否超过范围或不符合要求
10110	no license	无授权许可	检查参数值是否超过范围或不符合要求
10700	engine error	引擎错误	提供接口返回值，向服务提供商反馈
16003	basic component error	基础组件异常	重试或向服务提供商反馈
10800	over max connect limit	超过授权的连接数	确认连接数是否超过授权的连接数

四、价格套餐

	免费套餐	套餐一	套餐二	套餐三
服务量	200万字符	5000万字符	2亿字符	10亿字符
有效期	90天	一年	一年	一年
单价（万次）	免费	￥3528.00	￥1120.00	￥15960.00
立即购买	申请链接	购买链接	购买链接	购买链接