基于众包的深度学习数据标注系统

V1.0

# 项目背景

众所周知，深度学习(Deep Learning)算法在如今的人工智能领域，有着非常重要的地位。科大讯飞的语音识别、百度的自动驾驶、Face++的人脸识别等技术，都是基于深度学习构建的商用或未来可商用的人工智能系统。

对于深度学习来讲，除算法以外，数据是至关重要的! 如果我们把深度学习算法比作发动机，那么数据就是汽油，而原始的非结构化数据就是原油，在经过结构化之后的数据才能被深度学习算法利用。

数据结构化的过程，需要消耗大量的人工标注，而显然科研人员和工程师来做这种工作的成本是很高的。由于互联网用户体量巨大，所以以众包的形式解决数据标注问题是一个非常好的方式。

# 软件产品介绍

该系统是一个开放性、智能化的数据标注众包任务平台，任何组织或个人都可以注册并为深度学习提供数据服务，赚取一定金钱的同时实现人工智能时代的个人价值。该系统主要用于为在线标注的工作人员，提供用户登录注册、标注工作训练、标注任务分发、个人信息管理以及现金提现等功能。

## 系统核心：

依托任务分解、在线陪训、人力资源筛选等机制，提升数据标注的质量和效率。

# 软件架构介绍

系统分为前端页面展示和后端数据接口：

1. 前端：

采用Angular4.0版本，使用Typescript语言构建。组件化各个前端模块，使用Angular内置路由控制页面导航。

Angular(2.0以上)框架采用MVVM(Model-View-ViewModel)设计模式，默认采用强类型的、面向对象的Typescript语言，为组织、构建大型web应用提供了非常好的规范。

1. 后端：

采用基于node.js的express框架，为前端提供RESTful API接口；使用mysql数据库存储数据；使用sequelize数据库ORM引擎执行增删改查操作；用户认证采用JSON web token机制的中间件。

Node.js是一个异步事件驱动的javascript后端运行时。对于IO密集型的web后端应用来说，异步事件模型相比多线程模型有着很大的性能优势。

# 运行环境介绍

前端HTML页面和静态资源(css、javascript源文件和媒体资源文件)，通过Angular的脚手架编译构建后，存放在CDN服务器上，并通过nginx反向代理转发。

后端API接口服务，使用node.js的pm2工具，执行后端服务入口程序，创建多个cluster做热备和负载均衡处理，并通过nginx反向代理转发。

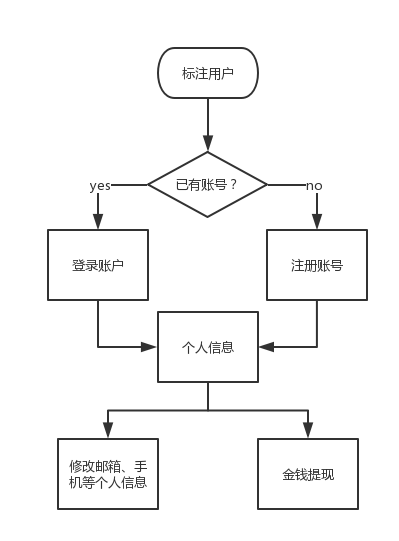
# 系统架构图

### 数据标注众包平台v1.0整体架构：

主要包含两个模块：用户管理和任务系统。

# 各子系统详情

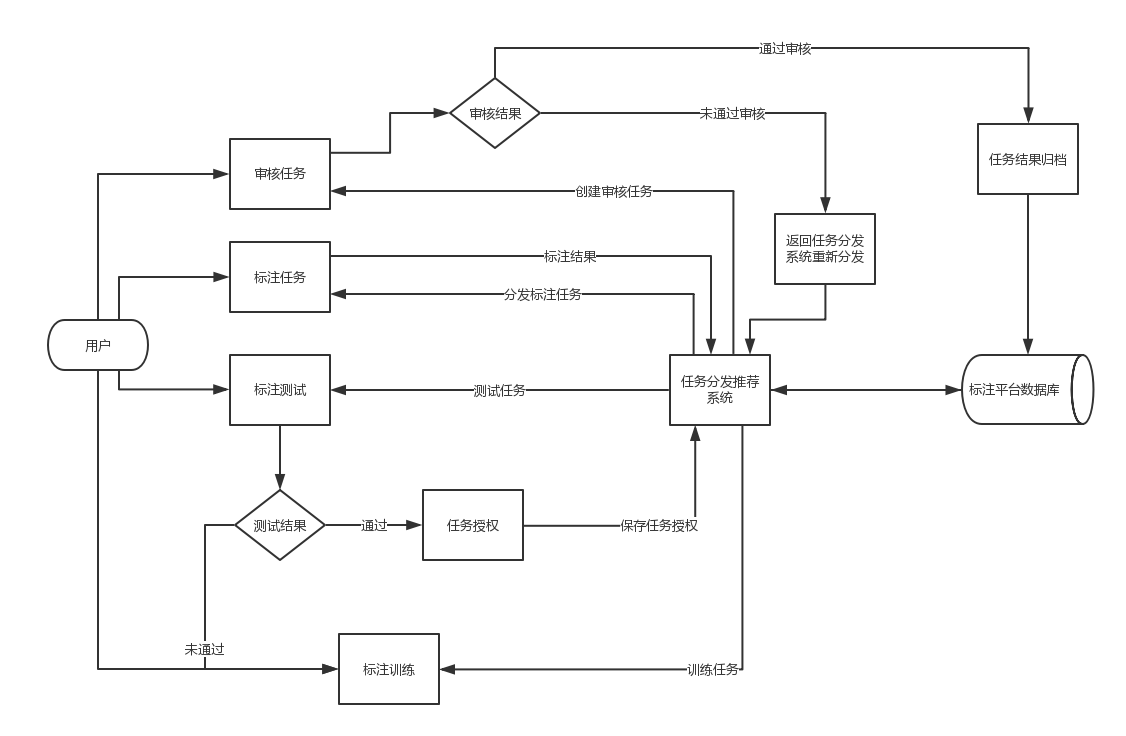
### 用户管理模块



用户管理流程图

用户管理模块主要功能有：用户注册和用户登录、个人信息的展示和管理、已赚取的金钱展示和提现。

### 任务系统



任务系统流程图

用户在注册系统后，没有权利获取任何实际的标注任务。只能首先通过一系列的标注训练，以增加相应技能的经验值，从而获取相应类型任务的授权。如：一个用户如果通过了图像画框的训练，并获取了足够的画框技能经验值，则获得了画框任务的授权，任务分发系统会根据每个用户已经获得的技能，为其分配对应类型的任务。

当有新任务到来时，任务分发系统会将标注任务分发到已经获取对应类型任务授权的用户端。当用户进入任务系统做该任务时，首先会经过一个测试阶段(quiz mode)。测试阶段中，包含一些已有正确答案的标注任务，如果用户能够在测试阶段拿到100%的准确率，则将该任务开放给当前用户进行标注。如果用户在测试阶段未能拿到100%的准确率，根据错误的数量，扣除一定的技能经验值，提醒用户需要重新进行一次该类型任务的训练，在通过训练之前，不再分发该类型任务给该用户。

当一个用户获得了一个类型任务的授权，并通过了测试后，用户即可开始进行真正的标注任务。每完成一个标注任务，系统会为该用户的标注结果自动创建一个审核任务。该审核任务会分配到另外一个已经获得该类型任务授权的用户手中，由其对前一个用户的标注结果进行审核。如果审核通过，则将前一个用户的标注结果标记为accept；否则，将该任务返回到任务分发系统中重新分发，而标注该任务的用户将会扣除一定的经验值和金钱。

# 部署说明

本系统可以部署在Windows、Linux、MacOS、UNIX等操作系统上，下面以Ubuntu系统为例，说明如何部署该系统：

1. 安装node.js运行环境：

访问 [https://nodejs.org/en/download 下载node.js的linux](https://nodejs.org/en/download%20下载node.js的linux)源码包。

在shell中，执行tar -xvf node-v\*\*\*.tar.gz将源码包解压后，切换到node-v\*\*\*目录。

执行./configure && make && sudo make install命令编译安装nodejs。

1. 安装typescript编译器：

安装好node.js运行环境后，在shell中执行sudo npm install typescript -g命令安装typescript编译器。

1. 安装和配置mysql数据库：

在shell中，运行sudo apt-get install mysql-server，安装mysql数据库。安装过程中，需要设置数据库root用户的密码。安装完成后，在mysql的shell中创建一个新的数据库，并记录该数据库的名字。如：create database testdb; 创建的数据库名为testdb。

在后端的源代码中，找到./db目录下的setup.ts，修改其中的db\_config变量如下：

const db\_config: any = {

host: "mysql数据库服务器地址",

protocol: "mysql",

database: "testdb",

username: "root",

password: "root用户的密码"

};

1. 编译server代码并运行服务器：

切换到server代码目录，执行tsc命令，编译生成可执行js文件。

执行sudo npm install pm2 -g命令，安装pm2集群服务环境。在server代码目录下，使用pm2 start ./build/index.js --name api -i 4 命令运行编译好的index.js。此时api服务已经启动，监听端口为3001。

1. 安装angular/cli脚手架，编译打包前端代码：

执行sudo npm install @angular/cli -g命令，安装angular脚手架。切换到前端代码根目录，执行ng build --aot --prod 命令编译打包前端代码。将./dist目录下编译打包好的所有html、CSS、javascript源文件以及媒体资源文件，上传到静态服务器。

1. 安装nginx并配置静态服务器代理和api网关代理：

执行sudo apt-get install nginx安装nginx服务。修改位于/etc/nginx目录下的nginx.conf配置文件如下：

server {

listen 80;

server\_name localhost;

location / {

proxy\_pass “静态服务器的index.html”;

proxy\_connect\_timeout 1;

proxy\_send\_timeout 30;

proxy\_read\_timeout 60;

}

location /api {

proxy\_pass http://localhost:3100;

proxy\_connect\_timeout 1;

proxy\_send\_timeout 30;

proxy\_read\_timeout 60;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root html;

}

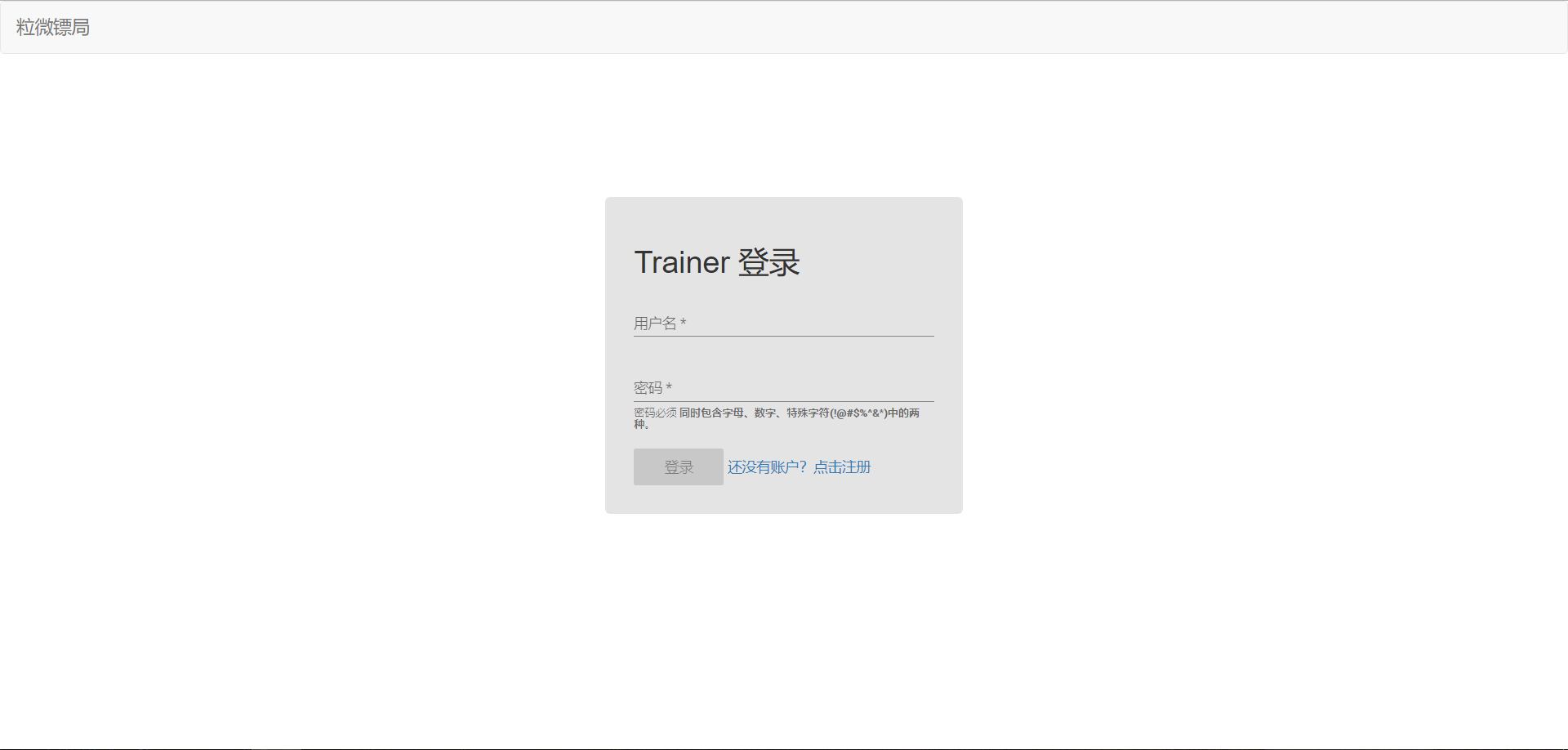
}

执行sudo service nginx restart重新启动nginx服务。

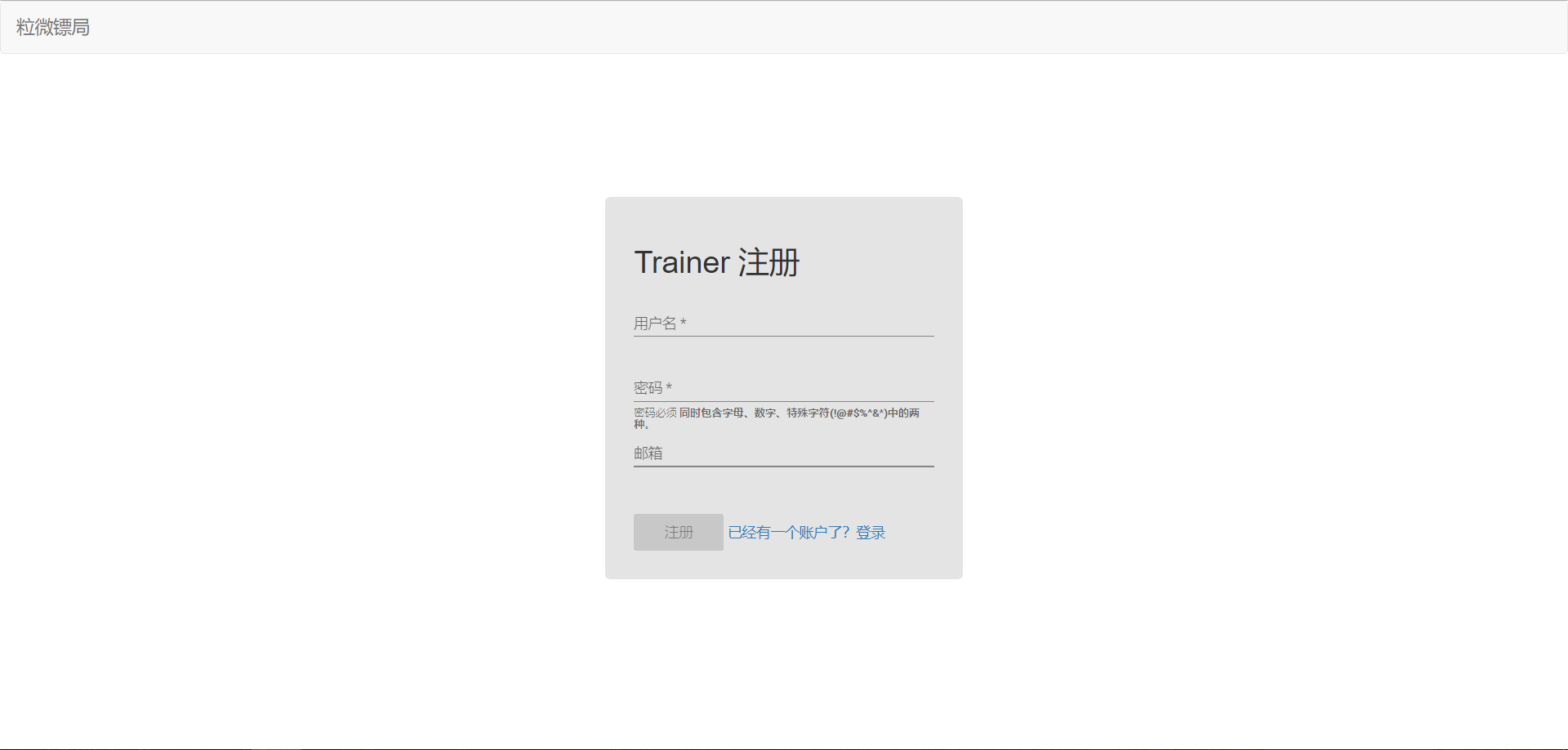
至此，就可以通过浏览器访问nginx服务器地址来查看该系统了。

# 用户使用手册

在浏览器中输入平台地址，如果之前未登录过或者登录用户信息过期，浏览器会自动跳转到登录页面。



如果是第一次登录本平台，请点击“还没有账户？点击注册”链接，跳转到用户注册页面：



根据要求输入用户名、密码和邮箱后，点击注册按钮

