<!DOCTYPE html>  
<html lang="en">  
 <head>  
 <meta charset="UTF-8" />  
 <link rel="icon" href="./public/1.ico" />  
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
 <title>文本训练平台</title>  
 </head>  
 <body>  
 <div id="app"></div>  
 <script type="module" src="/src/main.js"></script>  
 </body>  
</html>  
import { createApp } from 'vue';  
import './style.css';  
import App from './App.vue';  
import router from './router';  
import ElementPlus from 'element-plus';  
import \* as ElementPlusIconsVue from '@element-plus/icons-vue';  
import '../node\_modules/element-plus/dist/index.css';  
import { createPinia } from 'pinia';  
import piniaPluginPersistedstate from 'pinia-plugin-persistedstate';  
const app = createApp(App);  
const pinia = createPinia();  
pinia.use(piniaPluginPersistedstate);  
for (const [key, component] of Object.entries(ElementPlusIconsVue)) {  
 app.component(key, component);  
}  
app.use(ElementPlus);  
app.use(router);  
app.use(pinia);  
app.mount('  
<script setup>  
import zhCN from "element-plus/dist/locale/zh-cn.mjs";  
import { onUnmounted } from "vue";  
onUnmounted(() => {  
 localStorage.removeItem("token");  
})  
</script>  
<template>  
 <el-config-provider :locale="zhCN">  
 <router-view></router-view>  
 </el-config-provider>  
</template>  
<style scoped></style>  
import { createRouter, createWebHistory } from 'vue-router';  
import { useUserStore } from '../store/useUserStore';  
const layoutMap = [  
 {  
 path: 'model',  
 name: 'model',  
 meta: { title: '模型列表' },  
 component: () => import('../pages/model/modelList.vue'),  
 },  
 {  
 path: 'model/info',  
 name: 'modelinfo',  
 meta: { title: '模型详情' },  
 component: () => import('../pages/model/modelInfo.vue'),  
 },  
 {  
 path: 'model/info/training',  
 name: 'training',  
 meta: { title: '模型训练' },  
 component: () => import('../pages/model/modeltrain.vue'),  
 },  
 {  
 path: 'model/info/trainhistory',  
 name: 'trainhistory',  
 meta: { title: '训练记录' },  
 component: () => import('../pages/model/modelTrainHistory.vue'),  
 },  
 {  
 path: 'project',  
 name: 'project',  
 meta: { title: '项目列表' },  
 component: () => import('../pages/project/projectList.vue'),  
 },  
 {  
 path: 'project/projectinfo',  
 name: 'projectinfo',  
 meta: { title: '项目详情' },  
 component: () => import('../pages/project/projectInfo.vue'),  
 },  
 {  
 path: 'project/projectinfo/forecastdata',  
 name: 'forecastdata',  
 meta: { title: '预测数据' },  
 component: () => import('../pages/project/forecastData.vue'),  
 },  
 {  
 path: 'system/manageprojectcol',  
 name: 'manageprojectcol',  
 meta: { title: '项目列设置' },  
 component: () => import('../pages/system/manageProjectCol.vue'),  
 },  
];  
const routes = [  
 {  
 path: '/login',  
 name: 'login',  
 meta: { title: '登录' },  
 component: () => import('../pages/login.vue'),  
 },  
 {  
 path: '/',  
 component: () => import('../pages/layout/home.vue'),  
 redirect: { name: 'project' },  
 children: [...layoutMap],  
 },  
];  
export const router = createRouter({  
 history: createWebHistory('/h5/'),  
 routes: routes,  
});  
router.beforeEach((to, from, next) => {  
 let token = localStorage.getItem('token');  
 if (token == null || token == '') {  
 if (to.path == '/login') {  
 next();  
 } else {  
 next({ path: '/login' });  
 }  
 } else {  
 if (to.meta.title) {  
 document.title = '文本训练平台-' + to.meta.title;  
 }  
 if (to.path.indexOf('model') != -1) {  
 useUserStore.defaultActive = '/model';  
 }  
 if (to.path.indexOf('project') != -1) {  
 useUserStore.defaultActive = '/project';  
 }  
 if (to.path.indexOf('system') != -1) {  
 useUserStore.defaultActive = to.path;  
 }  
 next();  
 }  
});  
export default router;  
import { defineStore } from 'pinia';  
export const useUserStore = defineStore('useUserStore', {  
 state: () => ({  
 user: {  
 fUserId: '123456',  
 fUserTitle: 'admin',  
 fUserName: '',  
 },  
 defaultActive: '',  
 }),  
 getters: {},  
 actions: {},  
 persist: {  
 storage: localStorage,  
 },  
});  
from \_\_future\_\_ import absolute\_import  
from \_\_future\_\_ import division, print\_function, unicode\_literals  
import itertools  
from copy import deepcopy  
class Ngrams(object):  
 def \_\_init\_\_(self, ngrams={}, exclusive=True):  
 if exclusive:  
 self.\_ngrams = set(ngrams)  
 else:  
 self.\_ngrams = list(ngrams)  
 self.exclusive = exclusive  
 def add(self, o):  
 if self.exclusive:  
 self.\_ngrams.add(o)  
 else:  
 self.\_ngrams.append(o)  
 def \_\_len\_\_(self):  
 return len(self.\_ngrams)  
 def intersection(self, o):  
 if self.exclusive:  
 inter\_set = self.\_ngrams.intersection(o.\_ngrams)  
 return Ngrams(inter\_set, exclusive=True)  
 else:  
 other\_list = deepcopy(o.\_ngrams)  
 inter\_list = []  
 for e in self.\_ngrams:  
 try:  
 i = other\_list.index(e)  
 except ValueError:  
 continue  
 other\_list.pop(i)  
 inter\_list.append(e)  
 return Ngrams(inter\_list, exclusive=False)  
 def union(self, \*ngrams):  
 if self.exclusive:  
 union\_set = self.\_ngrams  
 for o in ngrams:  
 union\_set = union\_set.union(o.\_ngrams)  
 return Ngrams(union\_set, exclusive=True)  
 else:  
 union\_list = deepcopy(self.\_ngrams)  
 for o in ngrams:  
 union\_list.extend(o.\_ngrams)  
 return Ngrams(union\_list, exclusive=False)  
def \_get\_ngrams(n, text, exclusive=True):  
 ngram\_set = Ngrams(exclusive=exclusive)  
 text\_length = len(text)  
 max\_index\_ngram\_start = text\_length - n  
 for i in range(max\_index\_ngram\_start + 1):  
 ngram\_set.add(tuple(text[i:i + n]))  
 return ngram\_set  
def \_split\_into\_words(sentences):  
 return list(itertools.chain(\*[\_.split(" ") for \_ in sentences]))  
def \_get\_word\_ngrams(n, sentences, exclusive=True):  
 assert len(sentences) > 0  
 assert n > 0  
 words = \_split\_into\_words(sentences)  
 return \_get\_ngrams(n, words, exclusive=exclusive)  
def \_len\_lcs(x, y):  
 table = \_lcs(x, y)  
 n, m = len(x), len(y)  
 return table[n, m]