![A picture containing text, clipart

Description automatically generated](data:image/jpeg;base64,/9j/4AAQSkZJRgABAQEAYABgAAD/4RD0RXhpZgAATU0AKgAAAAgABAE7AAIAAAAUAAAISodpAAQAAAABAAAIXpydAAEAAAAWAAAQ1uocAAcAAAgMAAAAPgAAAAAc6gAAAAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAANeQ15XXqNeZ16og15PXnteR15UAAAWQAwACAAAAFAAAEKyQBAACAAAAFAAAEMCSkQACAAAAAzM1AACSkgACAAAAAzM1AADqHAAHAAAIDAAACKAAAAAAHOoAAAAIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAyMDIyOjA1OjE1IDE1OjA0OjU2ADIwMjI6MDU6MTUgMTU6MDQ6NTYAAADQBdUF6AXZBeoFIADTBd4F0QXVBQAA/+ELJmh0dHA6Ly9ucy5hZG9iZS5jb20veGFwLzEuMC8APD94cGFja2V0IGJlZ2luPSfvu78nIGlkPSdXNU0wTXBDZWhpSHpyZVN6TlRjemtjOWQnPz4NCjx4OnhtcG1ldGEgeG1sbnM6eD0iYWRvYmU6bnM6bWV0YS8iPjxyZGY6UkRGIHhtbG5zOnJkZj0iaHR0cDovL3d3dy53My5vcmcvMTk5OS8wMi8yMi1yZGYtc3ludGF4LW5zIyI+PHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9InV1aWQ6ZmFmNWJkZDUtYmEzZC0xMWRhLWFkMzEtZDMzZDc1MTgyZjFiIiB4bWxuczpkYz0iaHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbnRzLzEuMS8iLz48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOnhtcD0iaHR0cDovL25zLmFkb2JlLmNvbS94YXAvMS4wLyI+PHhtcDpDcmVhdGVEYXRlPjIwMjItMDUtMTVUMTU6MDQ6NTYuMzQ1PC94bXA6Q3JlYXRlRGF0ZT48L3JkZjpEZXNjcmlwdGlvbj48cmRmOkRlc2NyaXB0aW9uIHJkZjphYm91dD0idXVpZDpmYWY1YmRkNS1iYTNkLTExZGEtYWQzMS1kMzNkNzUxODJmMWIiIHhtbG5zOmRjPSJodHRwOi8vcHVybC5vcmcvZGMvZWxlbWVudHMvMS4xLyI+PGRjOmNyZWF0b3I+PHJkZjpTZXEgeG1sbnM6cmRmPSJodHRwOi8vd3d3LnczLm9yZy8xOTk5LzAyLzIyLXJkZi1zeW50YXgtbnMjIj48cmRmOmxpPteQ15XXqNeZ16og15PXnteR15U8L3JkZjpsaT48L3JkZjpTZXE+DQoJCQk8L2RjOmNyZWF0b3I+PC9yZGY6RGVzY3JpcHRpb24+PC9yZGY6UkRGPjwveDp4bXBtZXRhPg0KICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIAogICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgICAgICAgICAgICAgICA8P3hwYWNrZXQgZW5kPSd3Jz8+/9sAQwAHBQUGBQQHBgUGCAcHCAoRCwoJCQoVDxAMERgVGhkYFRgXGx4nIRsdJR0XGCIuIiUoKSssKxogLzMvKjInKisq/9sAQwEHCAgKCQoUCwsUKhwYHCoqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioqKioq/8AAEQgAywNGAwEiAAIRAQMRAf/EAB8AAAEFAQEBAQEBAAAAAAAAAAABAgMEBQYHCAkKC//EALUQAAIBAwMCBAMFBQQEAAABfQECAwAEEQUSITFBBhNRYQcicRQygZGhCCNCscEVUtHwJDNicoIJChYXGBkaJSYnKCkqNDU2Nzg5OkNERUZHSElKU1RVVldYWVpjZGVmZ2hpanN0dXZ3eHl6g4SFhoeIiYqSk5SVlpeYmZqio6Slpqeoqaqys7S1tre4ubrCw8TFxsfIycrS09TV1tfY2drh4uPk5ebn6Onq8fLz9PX29/j5+v/EAB8BAAMBAQEBAQEBAQEAAAAAAAABAgMEBQYHCAkKC//EALURAAIBAgQEAwQHBQQEAAECdwABAgMRBAUhMQYSQVEHYXETIjKBCBRCkaGxwQkjM1LwFWJy0QoWJDThJfEXGBkaJicoKSo1Njc4OTpDREVGR0hJSlNUVVZXWFlaY2RlZmdoaWpzdHV2d3h5eoKDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uLj5OXm5+jp6vLz9PX29/j5+v/aAAwDAQACEQMRAD8A+kaKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigArntd8eeGvDWoCy1vVY7S5KCQRsrE7SSAeAfQ10NfNH7QyMPiJbMVIVrCPBx1+d63oU1UnysTPpSCaO4t454W3xyKHRh3BGQafXi+mftCeHrPSbS2m0rU/MhgSNiojIyFAOPm6cV1+p/FXSrDwhZeJrewvr7TLpijSQKuYWBxtcEjHIIz049xSlQqJ2sFzuaK5bwR8QdH8eWc0ulGSGa3bEttOAJFB6NwTkH1rF8W/GTRvB/iSXRtQ0+/lmiVWMkIQqQwBGMsD3qVSm5cttRnodFcT40+KOk+CP7P+32l5ci/iMsRtwvAGOuSP7wq54H+IWkePba5k0lZ4ZLUqJYbhQGAOcHgkEcGj2c+XmtoB1VFed6Z8ZtD1LxkvhsWN9b3LXLWwkmVAm8Ejs2eSMdO9N8WfGvw74W1iTTPJudQuYTtm+zhQsbf3ck8n6U/Y1L2sFz0aiuK8E/FTQfHF09nYie1vVUv9nuAAWUdSpBIP8AOubuv2hfDtpeTW76VqjNDIyEhY8Eg4/v0KjUbtYVz1miuA8IfGLw54v1dNMtkurO7kB8pLlVAkwM4BBPOO1QeKvjTonhLxFcaPfafqE01vt3PCqFTkA8ZYHvR7GpzcttR3PRqK878M/Gzwt4k1KKwBudPuJmCRC7QBXY9BuBIBPvV3xz8U9K8B6nb2WpWV5cSXEPnK1uFIAyRg5YelHsqnNy21A7eivIl/aM8NFwG0vVFBPJKR8f+P16ZoOu2HiTRYNV0mbzrWcZU4wQRwQR2INKdKcFeSA0aKhu7lbOynuZAWSGNpGC9SAM/wBK8n/4aN8N/wDQJ1X/AL5j/wDi6IU5z+FAevUV5RY/tC+Fbq8SG4tNRtEY4M0kaFV+u1if0ruPFXjPSfCHh9dX1OR3t5WVYVhG5pSQSMdugJzQ6U4tJrcDforyD/ho3w3/ANAnVf8AvmP/AOLq3pn7QPhbUNRitZrXULNZWCiaZEKqT64YnFX9Xqr7Iro9Uorh/HHxU0nwJqNtZ6jZ3ly9xD5ytbhCAMkd2HpXMf8ADRvhv/oE6r/3zH/8XSjRqSV0guev0VxngT4maZ4+uLyHTLO7t2tEVnNwFAO4kcYJ9K5ub9oPw3balJaT6dqa+VKY2k2IQMHBP3s0vY1G2rbBc9XornNQ8caTZ+BW8WQGS804Irr5IG5gWC4wSMEE8g+lcJ/w0b4b/wCgTqv/AHzH/wDF0Ro1JbIZ69XN6r8QfC+iau2l6pq0VveLt3RMjEjcMjoMdCK5rw58cfDHiHWIdN8u8sJZ2CRNcouxmPQZUnGfevIvjSzWvxhup2QkKsDgHjcAi/4VrSw7lPlnoK59TA5GRRXkEf7RnhvCB9K1RegJ2xkD/wAer1LR9WtNd0e11PTnL211GJI2IwcH1HY1jOlOHxIZdoqtqN6mm6ZdX0ys0dtE0rKvUhRk4/KvKv8Aho3w2P8AmE6r/wB8x/8AxdEKc5/CgPXqK8iX9ozw0WAbS9UUE8kpHx/4/XbP4/0ZvAc3iyyaS70+FcssagSA7gpXBIwQTTlRqR3QHT0V5r4b+OPhzxJr1tpMdrfWk104jiedF2lj0HyscZqa9+M+had40fw3eWV9HOl0LZp2VBGCSBu+9nbznp0o9jUvawrnolFcH41+LugeC9SGnXCXF7egBpIrYD90DyNxJ6+1R+DvjH4e8YaommwpcWN7Jnyo7gDEmBkgMCecdjij2VTl5raDPQKK878W/GbRfB/iSfRr/T7+eeAIWeFUKncoYdWB6GsT/ho3w3/0CdV/75j/APi6aoVGrpCuev0VxWg/E/S/EHg/VfEVrZ3kdtpe7zY5Au9tqhuMNjoe5rlv+GjfDf8A0CdV/wC+Y/8A4uhUajbSWwXPXqK8hX9ozw0WAbStVUE8nbHx/wCP12up+PbGx8Ew+KbOzu9R06RQ7G2Vd8an+JgxHAPB9KUqNSNroLnU0Vxfgf4o6H47uJ7bT1ntbqEbvIuQoZ17suCc471F43+Kul+A9WgsNTsL2d54RMjwBCuMkY5Yc8UvZT5uW2ozuaK4vX/ifpPh/wAI6X4intbue11Pb5SRBd65Utzkgdsda5X/AIaN8N/9AnVf++Y//i6qNGpJXSFc9eorh/BfxY8P+NtQawsBcWt4FLrDcoAXA64IJH4V3FZyjKLtJDCiuN8d/EvTPAM1nHqdnd3BvFdk+zhTjaRnOSPWuT/4aN8N/wDQJ1X/AL5j/wDi6uNGpJXSFc9dd1jRnc4VRkn0Fc7ovxA8MeItSFho2rRXV0ylhGqMCQOp5FUPCvxG0Tx5p96mkedHcwwlpLadQHAIxkYJBGfSvnL4deK7XwV42XVtQt554VikjKQgbst9SK1p4dyUk90Fz6/rm774heFtM1ptIvtXiivldY2hKMSGbGBkDHcVyug/Hjw1rmt22m/Zb6ze5cRxyzomzcTgA4YkZNeQ/FKc6f8AGy+u5I2Kw3EMu3puAVDx+VOlh3KXLPTQLn1ZRXkSftGeGmdQ+l6ogJ5O2M4/8frrfE/xK0fwz4Z07XZIri8s9RKiA24XOCu4EhiOwrF0aiaTW4XOworya2/aI8LTXCpPYanboesjRoQPwDZrtNa8daVpHgoeKIhJqGnHYVNqASQxxnkjGD1zQ6NSLSa3GdLRXF+GfiloHiXw9qOsKZrG200/6QLkDcoIyCApOc9B71ysn7Rfhxbooml6k8IbAlwgyPXG6hUajbSWwrnr1FYen+L9K1bwlL4i0yVriyiieRgBhxsGSpB6GvPf+GjfDf8A0CdV/wC+Y/8A4uiNKpK9lsM9eoryvTf2gvCt9fR29xbX9krsF86ZFKL9drEgV6mrB1DKQVIyCO9TOnKHxIBaKKKgCtf6jZ6Xa/aNRuYrWHeqeZKwUbmOAM+5qz16V83/AB78aHVfESeHbKT/AETTTmfaeHmI/wDZQcfUmtT4Q/F7yPI8OeKrj91wlpeyH7nojn09D26Gur6tL2fOhXPfKKAc9KK5RhRRRQAUUUUAFFFFABWdF4g0mbXpdFiv4W1KGMSPbBvmVf8APb3rgPiv8V4fCVs+k6JIkusyLhmHItQe5/2vQfiff5vttZ1G01pdWt7yZL9ZfNFxu+bd6k967KWFdSPM9Owrn3BRXA/DH4nWnjnTxbXZS31mBf30IOBKP76e3qO1d9XLKLg+WQwoooqQCiiigAooooAKbJIkUbSSsqIgLMzHAUDqSaJJEijaSVlREBZmY4CgdSTXzf8AFv4tv4ikl0Lw5KyaUh2zzqcG6I7D/Y/nWtKlKrKyEfRVhf2uqWMV7p86XFtMN0csZyrD2qxXgX7PvjMxXU/hW+k/dy5nsyx6N/Gg+o5/A+te+0Vabpz5RhRRRWQBRRRQAUUUUAFFFFABRRRQAVQ1PQtJ1nZ/a2m2t6Y/uG4hVyv0yOKv1Q1vWrPw9otxqupuyWtsAZGVSxAJA6D3Ipq99APGfj74b0XR/DOmXGk6XaWUrXZRngiCFl2E4OPcVqeAdS0fSv2dzceJIxNp5M6SQkZMpMhAUe5P5de1cJ8WfiLB8QLyw0vw9bzva28hZSyfPPIeBhRzgD8eau/EbRr3wp8GvC2i3AZHaeSa5XPSQgsFP03H8q9JQbpwhPdsk4LQNev/AAj4hg8Q6NDLDbec6xrISVlQEboy3fgjP4Gui8TXDfFj4rRnw7BKUuUhj+Zf9WoUb2b2BJ59q9C8C+DbHxn8AYtMl2rO000sM2OYpgxAP0xgH2rhvhZ4pk+Hnjy40nXYlgguZPs10zr80Lg4DZ67c9e2DntWvOpOUor3loB7L8VND02X4YalJcWcU81hZ4tppEBePGOh7dK4D9mz/j81/wD65w/zevctQ0+01fTZrHUIVuLW4XbJGx4cfhWfoPhHQfDDzPoOmxWTTgCUxkncBnHUn1NefGqlScH1H1Pnv416HL4X+JCavp7eUt9i7iZTykqkbv1wfxr0n4XfD/RbvwNFqniDToNRv9X3TzS3KbjhicAZ6cc5HPNeefHi+k1b4mW+mQfObWCOFVB/jc7v/ZgPwr1nWviH4f8AhtHp+g6ml00kNlHs8iIEFQNo6kc/Ka6ZubpQit2LqeI/DqFbD46Wltalkiivp4lGf4QHAH5Cvo+TwR4Xldnk8P6azOSWY2yZJPU9K+cPhnK2q/G+zvLaJ9kt3NPgjlVIc8/nX1ZUYttTXoNHyL4ZjWx+NNnDaDyo4dYMaKP4VEhAH5V9SX3hTQNTu3utR0axurh8bpZbdWY9upFfLuh/8lwg/wCw2f8A0aa988SfGDw34W16fSdTW8NzBt3mKIFeQCOc+9XiYzlKPLvYSPEvjP4csvC3j5F0WAWcE9uk6xxnARskHHpyM19EWek6R4p0HStS1nTLS+mlsonDzwq5G5Q2Bkepr5s8d+In+J3xChfRLSULIEtbWJ/vNyeTjpyT9BXvWu/EDQvhvBpmjawLppFs02GGMMNqjZ6j+7SrRm4Qj9oEee/H7wno+j6TpOoaRp1vZSNO0Mn2dAgcbdwyB6YPPvXU/s+NIfhzKHzsF6+zP+6ua84+L/xM0rxzaadZ6LDcLHbSNLJJOoXJIwAACfevT/g14j8PXPh+Lw7oH2hpbCAS3EksW0O7H5iOfU/liioprDpSWodT0W8khisZ5LoZgSNmkBGcqBzx34rznwxqfww8XasdO0TRbKW5WIylX04INoIB5I9xXfa3/wAi/qP/AF6y/wDoBr5z/Z7/AOSjzf8AXhJ/6ElYUoJ05SvsMX49aBpWheKdP/sexhsluLXfIkK7VLBiM46Dit34qEt8CvBpYkkpb5J/69zVP9o//kaNI/68z/6Ga3fH+h3er/s+eHZrGNpWsLa2nkRRk7PJ2k/hkH6ZrqjL3abYi18E/Cegal8O0vdR0izu7mW5kDSTxBzgHAHPQV5r8a9H0/RPiK9vpNpHaQvbRyGOJdq7jkEgduldl8GviZ4f0Dwm+i69dfYpYZnljkZCVdWwccA4IOa4nxpqn/Cz/iqo0CGSSOYx20GV5ZR1cjsOSfpVU1NV5OWwdD6Vfw7o2s2dnLq2l2l7IkCqrzwq5Ax05rxD4/6dpui3GjWekaZZ2SSpJLI0ECozkEAAkDp1/OvoWCLyLeOIHPloFz9BivAf2kv+QzoX/XvL/wChCuXCtuqkN7Hp3wq0fTtP+Huj3VlZQw3F1ZxvPKiANKevzHvyTXn/AO0B4d0fTPD+nXunabbWtzLdsskkMYUuCpJzjrzXpHgC4S0+E+i3MudkOmpI2BzgLk14z8XviZonjbQrGy0ZLoSQXBlczRhQBtI9T61VFTde67h0O1+Gl/pGm/AUXXiSNZtNjmk85Hi8wEGTA+XvyRW54UPw48afaf7B0Oxl+zbfM8ywCY3Zx1HPQ1wemf8AJqd9/wBdW/8AR61Y/Zs/1evfWH/2aqnD3ZzvrcDg/Hel2Wi/Gqey0uBba2ju4GSJOAu5UY49OSa+oNS8PaNrEqSatpVneSINqvPArkD0yR0r5q+J/wDyXm7/AOvm2/8ARcdfSPiPxJp3hTR21TWZHjtVdULIhY5JwOBSr8zjTtvb/IEeG/tAaBpOitoraRp1tZGUSh/s8YTdjbjOPqa9Y+FH/JK9B/69/wD2Y14R8TfGh+JnimxtPD1ncPBADFboV+eZ2PJ2jp0FfRvg/RW8O+DtL0mUhpLW3VJCOhbq36k0VrxoxjLcFua0sUc8LxTIskcilXRhkMD1BFfLHiC2tJvj0NPFjbRWUepxW4toogqFAyjBA655z9a+lvEniCz8L+H7nWNSEhtrbbvES5b5mCjA+pFfMWmXx8YfHS11DToJFS71VJ1RhllQMCScegBNGFTtKXSwM9T+NPhDw9p/w7mvtP0eztLmGeMJLBEEIBbBHHUVW/Z9tYNQ8C6xZ30KXFs94N0Mq7lb5F6g/QV0Xx1/5JXef9d4f/Qqwv2cf+RS1X/r9H/oApJt4Zt9w6nmNvawWPx/gtrOJYYIteRI40GAoEwwBXTftCeGzp/iSz8Q2w2pfJ5cuOMSoOD+K4/75r2f/hX/AIWOtjWP7Gg/tAT/AGjz8tnzM7t3XGc8149+0dqvm67pGko3EEDTuvu7YH6KfzrWnV9pVjy9hdDovg94PsNf8P3HinxRaR6pqGpTOA92gcBF+XIB4ySDz7CvLp7ODQvjulrpaeRBba0ixIDnaPMHH0r2mDxjovwt8FeHNM1pLjzJrIMPIjDc4BbPI7tXidvfp4q+N1vf6bHII73V0ljVx8wXeDk49hmqpOTlOT2A+oL/AMK6Bql411qOjWN1cOAGlmgVmOBgckelfOXxvs7LT/iBDp+m2FtZW0VrGQlvEE3FiSScdT0H4V9R18w/Hn/kqh/69Yf61hhG3UsNnuxsdI8M/Dy8ltdItjaxWDTzWqIFWfbHkhuOcgYyc1wXw68QeE/H+tXOnjwJptj5EBm8zakm75gMY2D1r0DxN/ySvVf+wPL/AOiTXh3wDvoNM8Ra3f3ZK29rpbzSkDJCqyk8d+BSpx5qcpdQNb9oDw7o+jWOjS6TplrZPJJIrm3iCbgAMZxXefDa9sdN+B9leasyLZQ28rT7xkFdzZGO+emO+a8j+KfjhPiV4g03TvDVrcSw25ZIgU+eeRyOQvYYA/Wu4+Iek3PhT9nmw0dm2yo8KXOw8Ekl2H03fyrWUW6cKct2xdTxltTk03xPL4i8J29xYWcF3m2LHcI85IRj05APHp610fj7xOfij4j0IaNZyNeNaJBJAAf9cWJIH+zyDmu3+Cvh3TvE/wALNe0vUUDLc3pVj/EhEalGHuCSa4fwzqV78IvijJBq0KvHGxt7k7MlomIIkQ9emG9+ldHMnJ2XvRA9w1dvC3gjwHoth4zgiuba3VII99v5w80JyQMcd6saBovgLxdoS6jpOgafJZzFkDNZqhODg9siuO/aGniuvAujXFtIssMt4HR1OQymNiCK6D4Ff8kpsv8ArvN/6Ga4XFqj7S7vcfU8Z+G8CWfx0tLa3ysUV7PGoz/CA4Ar6rr5Y8A/8l+g/wCwhcfyevqenjPjXoCOK8feIPBeizWS+NbSK4eVXNv5lp52ACN3bjqKr3fhrwXrvgC61TTdCsfs9xYyTQyC1EbjCkg9Mg5FcB+0n/x/aB/1zm/mld/4R/5INaf9geT/ANAaly8tKM09w6nkP7PrEfEiQA4BspAR68rX0BN4M8M3M7zT6BpsksjbndrVCWPqeK+fv2fv+Skv/wBeUn81r23xp8S9C8Dyrb6r9oe6kh86KGGPO8ZI+90HINXiVJ1rRBbHgHiLT7TS/jz9j0+BLe2j1WDZEgwq5KHgfU19N6j4b0TV7gT6ppNleTAbRJPArtj0yRXzZ4I0/UPiL8XhrMluVgW8+23Lj7sSg5Vc+vAAr6P8TeKNM8I6T/aWtSPHbeYI8pGXO45xwPpRib3jFb2BHg/x+0HStF1bRzpGn29l50L+YIIwgbDDGQPrXrfgfR9O1n4W+HYdWsbe9jS0RlSeMOFOMZANeEfEDxRcfFTxxaQeH7KZ4418i1jI+d8nJY+n9AK93vvEGlfCvwTo1vrHnPHHGlqDAm4lwmSevTg06qkqcIfaA4/42eB9AsvAb6tpemW1ldWs0Y328YTcrHaQQOD1B/CqXwWtB4o+FWv+Hrx/3LSskZP8G9AQfwYZrI+Knxf0nxb4U/sXQoLrM0yvNLOgUBV5wBk5OcV1vwW0+Xwv8Lb/AFnUIzGLhnu1VxgmNE4P44OKb540LS3voLqeKeHNJ1C58Xjwg87wR3t4lvexoeG8tzn8sEivZvjD4H8OaT8L5bnTNJt7SeykiEUsSAMQXCkMercHvXDfAyyfWfipNqc43fZoZbhif77nb/7MT+FbvxQ+Lnh/xP4LvNE0yK8NzLKnzSxhVUK4Y9/atanO60VHpa4dC/8AAO5gg+H/AIhk1D57OGYvKhXcNnl/Nx34HSuk8LX3wy8Y6jLZaHotlLPFEZWD6eEAXIHUj1Irkfg5DInwh8WSuhWORZQjEcNiHn+dY/7On/I8aj/14H/0YtZ1IJupK+wGZ8c9C0zQvG8CaRZxWcc1qsjxxLtXduYZA7dBX0h4bJbwrpRYkk2cWSf9wV8//tE/8j1Zf9eK/wDobV7/AOGv+RU0n/rzh/8AQBWdZt0YNjW5p1ieMfEMfhbwhqGrykZt4j5YP8Uh4UfmRW3Xhv7RviDZa6XoEL/6xjdTAHsPlQfqx/CuejDnmogzwi5uJbu6luLhy8srl3Y9WJOSajpKK90k9y+EPxe8jyPDniq4/dcJaXsh+56I59PQ9ule+A5GRXwlXuHwh+L32fyPDniq4/dcJaXsh+56I59PQ9ulefiMN9uA0z32igHIyKK80oKKKKACvL/iv8V4fCVs+k6JIkusyLhmHItQe5/2vQfifc+K/wAV4fCVu+k6JIkusyLhmHItQe5/2vQfiff5muLia7uJLi5kaWaRizu5yWJ6kmu7D4fm9+ewmwuLia7uZLi5kaWaVizu5yWJ6kmoq1/DfhnU/Fesx6bo1uZpnOWbosa92Y9gK+mdC+D/AIb0vwbNol5breSXag3N2ygSFuxQ/wAIHb9c121a8KWjJsfLOnajd6TqMN9p0729zA4eORDgqa+pfhj8TrTxzp4trspb6zAv76EHAlH99Pb1Havnzx94B1HwJrRt7oGaylJNtdhflkHofRh3Fc5p2o3ek6hDfadO9vcwOHjkQ4KmlUpwrwug2PuWiuA+GPxOtPHOni2uylvrMC/voQcCUf309vUdq7+vHlFwfLIsKKKKkApskiRRtJKyoiAszMcBQOpJpXdY0Z3YKqjJYnAA9a+cPi/8WG1+aTQPDk5XSkO24nQ4Ny3oD/c/n9K1pUpVZWQhvxb+Lb+I5JdC8OSsmkods86nBuiOw/2P515HRRXtQhGnHliSXNJ1O40bV7XUbJylxayrKh9wc/lX2noerQa9oNlqlr/qruFZVHpkcj8DkfhXxBX0f+zz4gN94UvNGmfMmnzb4wT/AMs35/Rg35iuXGQvDm7DR6/RRRXlFBRRRQAUUUUAFFFFABRRRQAVW1HTbPV9PlsdSt0ubWYASRP0YZz/ADFWaKNgMXSPBvhzQZvO0fRrO0l/56JENw+hPIqxrXh7SfEdtHb65YQ3sUb70SUZCtjGf1rSoquaV73AoaPoem+H7E2ejWcdnb7y/lx5xk9TWbqngLwtrWoSX2q6Ja3N1JjfK6nLYGB+ldDRRzSTumBHb28VrbR29uuyKJQiLnOABgCpKKKkDAuPA3hm71o6tc6NbS35kWU3DAlt4xg9e2BUms+DPDviG8W61vSbe9nRNiySgkhc5x+prboqueXcDI0XwnoPh13fRNJtbJ5BhnijwxHpnrj2rXoopNtu7A52LwB4Vg1UalFodql4svnCYKdwfOd3Xrml1TwH4X1rUJL7VdEtbq6kxvlkUktgYHeuhop88r3uBjaN4Q8P+H5Wl0XR7SzlYYMkcY3Y9M9ah8QeB/Dvim7iude01LuaJPLRmdlwuc44I7k1v0Uc0r3vqBxX/Cn/AAL/ANAGP/v7J/8AFVr+H/BHh7wrcTT6DpyWksyBJGV2bcM5xyTW9RTdSbVmwGyxJPC8Uqh45FKsp7g8EViaN4K8OeH743mi6RbWdwUKGSJSDtPUfoK3aKlNpWQGNrfhDQPEdxHPrmlW97LEuxGlBJUZzitO1s7ezsYrO1iWO2hjEUcQHCqBgD6YqaijmbVgOM1D4S+CtSumuJ9Eijkc5byXaME/QHFa/h/wX4e8LbjoWlw2sjjDSgFnYem45NblFU6k2rNgFY+t+EtB8SSxSa7pcF88IKxtKCdoPUCtiipTad0BVt9Ns7TS0022t0js0i8pYVHyhMY2/TFc7/wq3wT/ANC3Zf8AfJ/xrrKKalJbMDLPhrRj4f8A7D/s6AaXjb9lC4TGc/z5puieF9E8N+d/YWmw2Pn48zygRux0/nWtRRzSta4GBf8AgbwzqmrNqmoaNbT3zMrNO4O4lQAD17ACtLVtG0/XtPax1i0ju7ZmDGKQZBI6GrtFHNLuBkaP4U0Hw+xbRdJtLNzwXiiAYj03da16KKTbbuwKmp6XZazp0thqlsl1azY8yKQcNggj9QKo6N4Q8P8Ah6ZpdE0i1s5XGGkjj+bHpnritmijmaVrgUtW0fT9d09rHV7SO7tXIZopBkEjkVFonh3SPDlvJBodhDZRStvdIhgMcYzWlRRd2sAVg6r4I8Na5qJv9X0e2u7ogL5sgJOB0HWt6ihNrYDH1rwloPiIwHW9Lt7026lYvNH3AcZA/IVHo/gvw3oF0bnR9GtLScjHmxx/MB7E9K3KKfNK1rgFYOr+CPDevah9u1jR7a7udoXzZFJOB0Fb1FJNrVAYPjRFj+HevIg2qumTgAdh5TV4V+z7bQ3vinWLW6jWWCfTWjkjbo6llBB/Cvo66tYL6zmtbuNZYJ0MckbdGUjBB/CsnRfB3h7w5dPc6HpVvZTSJsZ4gclc5x+lbwqqNOUe4iXR/Cug6AxbRtJtLNyMF4ogGI/3utWdW0bT9d09rHWLSO7tmYMYpBkEjoau0VhzO97jMvRPDWj+G4pY9C0+GySZg0ixAgMR0NV9Z8GeHPEN4t3rWkW15cKmwSSLzt64/WtyinzSve4GJdeDfD19o1tpN3pUE1haHMEDglYzz059zV7SdH0/QtPWx0i1jtLVCWWKMcAk5NXaKOZtWbAwLPwL4ZsNYGq2ejW0N8rtIJ1U7gxzk9fc1v0UUm29wMjW/CmheJHhbXdMgvmgBEZlBO0Hrj8hVy20qxtNJXS7a2jjsVjMQgUfKEPBH05q3RRzO1gMLR/BPhvw/ffbNG0e2s7jYU8yIEHaeo/Sn614P8P+IrqO51vSre9miTYjyqSVXOcfmTW1RT5pXvcCtYabZaVai20y0gtIB0jgjCL+QqHWNE03xBY/Y9Zs47y33B/LkGRkdDV+ild3uBlaP4Y0Pw+D/YulWtkW4LRRAMR6FutP1vw7pPiO2jt9csIb2KJ96JKMhWxjP5GtKijmd73A5i2+G3g20uFmg8OWAkU5BaLcB+B4roLuxtr6wlsruFZbaVDG8RHysvp9KnopuUnuwMfRPCeheG5Jn0LS4LF5wBI0QwWA6fzrMb4YeCncs3hyyLE5JKnk/nXV0Uc8r3uBTh0nT7fSjplvZww2RQxm3jQKm09RgetUNF8HeHvDt09zomk29lNImxniByVznH6Vt0UuZ9wMTWvBnh3xFdrda3pNvezomxXlBJC9cfrWxBBHbW8cECBIolCIg6KAMAU+ii7aswCvkn4waudX+KGqsGzHauLWP2CDB/8AHt1fWc0qwQSSv92NSx+gGa+H9Uu2vtXu7tzuaeZ5CfXLE/1ruwUfechMrwxPPMkUY3O7BVA7k19F+Mfgpa6n4NsP7DjSDWdPtEjPQC62ryG/2s5wfwPt4z8NtNGrfEjRLVl3IbpZGHsnzH/0GvsatMVVlCUeUSPhe6tZ7K6ltruJ4Z4mKSRuuGUjqCKir6h+Kvwqg8YWr6po6JDrUS/RbkD+Fv8Aa9D+B9vmO6tZ7K6ltruJ4Z4mKSRuuGUjqCK6aNaNWN1uJntfwh+L32fyPDniq4/dcJaXsh+56I59PQ9uhr30HIyK+Eq9w+EPxe+z+R4c8VXH7nhLS9kP3PRHPp6Ht0rlxGHv78BpnvteX/Ff4rw+E7d9I0SRJdZkXDMORag9z/teg/E+6fFf4sQ+E7Z9J0ORJtZlX5nHK2oPc/7XoPxPv8z3FxNd3Ek9zI0s0jFndzksT1JNZ4fD83vz2G2FxcTXdxJcXMjSzSMWd3OSxPUk1s+EfCGqeM9cTTtJiz3lmYfJCv8AeY/071Y8EeBtU8c6wLTTk2W8ZBuLph8kS/1PoO9fVvhTwnpfg7RU03R4dqD5pZW5eZv7zH/OK6q+IVJWW4kiDwX4J0vwRoq2WmR7pWANxcsPnmb1PoPQdq6KiivIlJyd2UZniDw/p3ifRZtL1iATW8o/4Eh7Mp7EV8oePvAOo+BNaNvdAzWUpJtbsL8sg9D6MO4r7CrM8QeH9O8T6LNpesQCa3lH/AkPZlPYiuihXdJ+Qmj4t0/ULvStQhvtOne3uYHDxyIcFTX1N8MfidaeOdPFtdlLfWYF/fQg4Eo/vp7eo7V8+eP/AADqPgTWjb3QM1lKSbW7C/LIPQ+jDuK5zT9Qu9K1CG+06d7e5gcPHIhwVNejUpwrwuidj7lpHdY0Z3YKqjJYnAA9a8/+GfxRsvGumGC/aO11e2TdPEThZFHWRfb1Hb6V5n8XPi42uPNoHhqYrpqnbcXKHBuT6D/Y/n9K82OHnKfIVcX4ufFxtcebQPDUxXTVO24uUODcn+6P9j+f06+O0VZ0/T7vVdQhsdOge4uZ3CRxoMljXrwhGnGyJDT9Pu9V1CGx06B7i5ncJHGgyWNfQVh8FrTR/hhq0N2sd1rt1alzKBkRMvzBE/EYJ7/Suj+GHwwtPA+ni6vAlxrU6/vZgMiIH+BPb1Pf6V6AQGUg9Dwa86vim3aGyKSPhKvSvgPq5034lRWrNiPUIHgI9SBuH/oNcV4p07+yPFuq2AXatvdyIo9FDHH6YqXwbfnS/G2jXgOBFexFj/s7gD+hNehNc9NruiT7TooorwSwooooAKKKKACiiigAooooAK5nXviJ4Y8M6n/Z+takLa52B9hjZuD0PA9q6avmD4//APJTT/15xf1rehTVSfKxM9m/4XJ4G/6Da/8Afl/8KP8Ahcngb/oNr/35f/Cvkqiu76nT7sVz61/4XJ4G/wCg2v8A35f/AAo/4XJ4G/6Da/8Afl/8K+SqKPqdPuwufWv/AAuTwN/0G1/78v8A4Uf8Lk8Df9Btf+/L/wCFfJVFH1On3YXPrX/hcngb/oNr/wB+X/wq7a/FHwVdsqxeIbQM3QSEp/MCvj2ij6lDuwufdFpe2t/AJrG5huYj0khkDqfxFTV8R6N4g1bw9erdaNfz2cqnrG5Ab2I6Eexr6U+FXxTTxvbtp+qCODWIE3EJws692UdiO4/yOWthZU1zLVDuekUUUVyDOKm+L3gm3uJIZtZVZI2KMPJfgg4Pao/+FyeBv+g2v/fl/wDCvlnXP+Rh1H/r6l/9DNUK9VYOnbdk3PrX/hcngb/oNr/35f8Awo/4XJ4G/wCg2v8A35f/AAr5Koo+p0+7C59a/wDC5PA3/QbX/vy/+FH/AAuTwN/0G1/78v8A4V8lUUfU6fdhc+tf+FyeBv8AoNr/AN+X/wAKP+FyeBv+g2v/AH5f/Cvkqij6nT7sLn1r/wALk8Df9Btf+/L/AOFH/C5PA3/QbX/vy/8AhXyVRR9Tp92Fz7a8P+JNL8Uae19odyLm3WQxlwpXDAA45+orUryv9nv/AJJzN/1/Sf8AoK16pXnVIqE3FFHI6n8UvCGjapPp+o6sIbq3fZInlOdp9MgVU/4XJ4G/6Da/9+X/AMK+d/il/wAlS1//AK+j/IVyVd8cJBxTuybn1r/wuTwN/wBBtf8Avy/+FKPjH4GLAf22oyephf8Awr5JoqvqdPuwufaWkeNfDWvSiLSdas7mU9IxJhz9FOCa3K+E0do2DIxVlOQQcEGvoL4I/Ey71i4PhnxBcNPcKhazuJDlnA6ox7kDkH0BrCtheSPNFjTPaqyvEPibSvC1gl7rl0LaB5BGrlS2WIJxx7A1q1j+KvDFh4u8Pz6TqiZjkGUcfeicdGHuP/rVxxtf3thnPf8AC5PA3/QbX/vy/wDhR/wuTwN/0G1/78v/AIV8zeLfCmo+DvEE2l6onzIcxSgfLMnZh/ng8Vh16SwlNq6bJufWv/C5PA3/AEG1/wC/L/4Uf8Lk8Df9Btf+/L/4V8lUU/qdPuwufbmheINL8S6aL/RLtLq3LFCy5G1h2IPINaVfIXw48fXXgXxCs4LS6dOQl3bg/eX+8P8AaHb8q+tdPv7XVNPgvrCZZ7a4QPHIp4YGuKvRdKXkNMsU2WVYYXlkOERSzHGcAU6kdQ6MrdGGDXOM4f8A4XJ4G/6DS/8Afl/8KP8Ahcngb/oNr/35f/CvljWrYWWvX9qowIbmSMD6MRVGvV+p031ZNz61/wCFyeBv+g2v/fl/8KP+FyeBv+g2v/fl/wDCvkqij6nT7sLn3FpOq2WuaXBqOlzie1nXdHIBjIzj+lXK8t/Z+vzdfDh7djn7HeSRgegIV/5sa9SrzakeSbiUcnq/xN8J6Dqs2napqqwXUBAkj8pztyM9QPQ1S/4XJ4G/6Da/9+X/AMK8C+Mf/JVtY/30/wDQFrh69CGEhKKbbJufWv8AwuTwN/0G1/78v/hR/wALk8Df9Btf+/L/AOFfJVFV9Tp92Fz61/4XJ4G/6Da/9+X/AMKP+FyeBv8AoNr/AN+X/wAK+SqKPqdPuwufX9v8VfBNyVCeIbVS3QSbk/mK6ax1Kx1OHzdNvLe7j/vQShx+lfDVXNN1bUNHvFutKvJ7SdDkPC5U/p1qZYKP2WFz7ioryX4TfF1vFMy6H4iKJqgUmCdRtFwAOQR2bHPHWvWq8+pTlTlyyKCuc8Q+PvDnhW+js9d1AW08kfmKpjZsrkjPA9jXR183/tF/8jxYf9eC/wDobVpQpqpPlYmer/8AC5PA3/QbX/vy/wDhR/wuTwN/0G1/78v/AIV8lUV3fU6fdiufWv8AwuTwN/0G1/78v/hR/wALk8Df9Btf+/L/AOFfJVFH1On3YXPrX/hcngb/AKDa/wDfl/8ACp4Pi34HuPu6/AnP/LRHX+Yr5Doo+pU+7C59wabrel6zH5mlahbXi4yfIlV8fUDpV6vhmzvrrT7lbixuZbaZDlZInKsPxFe6fC/41zXt5DofjCRWeUhLe/PGW7LJ25/vfn61z1cJKKvHUdz3KiiiuIZl+J5/svhPVp/+ednKwx/uGviY9a+zvHXmf8K/13yc7/sMuMf7pr4xr08F8LJZ6T8BbcT/ABSt3P8AywtZpB/3zt/9mr6kr5m/Z5/5KRcf9g6X/wBDjr6ZrDGfxRoK8z+Kvwqg8YWr6po6LDrUS/RbkD+Fv9r0P4H29MormhOUJc0RnwvdWs9ldS213E8M8TFJI3XDKR1BFQ19QfFj4Vx+L7RtV0WNI9ZhXkdBdKP4T/teh/A+3zJcW81pcyW9zE0U0TFXjcYKkdQRXs0aqqxuiBju0jlpGLMepY5Jrrfh/wDDzU/HerCO3DW+nxN/pN4y/Kg/uj1Y+n51yFegfC74mXHgbUvs15vn0a5cGaIcmJunmL7+o7iqqc6g+TcD6a8O+HdN8LaNDpmj26wwRjk/xSN3Zj3JrUqCyvbbUbGG8sZknt5kDxyIchge9T14Tbb1LCiiikAUUUUAZniDw/p3ifRZtL1iATW8w/4Eh7Mp7EetfKHj/wAAaj4E1o290DNZSkm1uwvyyD0Pow7ivsKvPfjH4i8P6V4NnsNdgS9uLxSLW0zht3aTPVQD3/CurDVZRlyrVMTPlVJHjbdG7IcEZU44PBFNoqzp2n3erahDY6dA9xczuEjjQZLGvYJDT9Pu9V1CGx06B7i5ncJHGgyWNfU3ww+GNp4G08XV4EuNanT99MORED/Ant6nvR8MfhjaeBtPFzeBLjWZ1/fTAZEQ/uJ7ep7139eViMRz+7HYpIKKKK4hnyR8YLX7L8VdZGMCSRZP++kBrjIXMc6OpwVYEH0rvvjjt/4WtqGzP+rizn18ta8+X7wr3qWtOPoQfdFrMLizhmHSSNXGPcZqWqmk/wDIFsf+veP/ANBFW68J7lhRRRSAKKKKACiiigAooooAK+YPj/8A8lNP/XnF/Wvp+vmD4/8A/JTj/wBecX/s1dmD/iCZ5hRS0V6xIlFLRQAlFLSUAFFFFABWp4b1yfw34ksdWtSQ9rMrkA/eX+JfxGR+NZdLQ0mrMD7otriO7tIbmA7o5kWRD6gjIqWuV+GV4198MtCmkOWFqIyf90lf6V1VfPyXLJos+INc/wCRh1H/AK+pf/QzVCr+uf8AIw6j/wBfUv8A6Gao1762IEopaKYCUUtJQAUUUUAFFFLQB9M/s9/8k5m/6/pP/QVr1SvK/wBnv/knM3/X9J/6CteqV4df+LIpHx98Uv8AkqWv/wDX0f5CuSrrfil/yVLX/wDr6P8AIVyVezT+BehIUUUVYBW34O1F9J8a6PexHDQ3kZP03AH9CaxKu6P/AMhyx/6+I/8A0IUmrqwH3DRRRXzxZyvj/wACWPjrw+1pcbYryIFrW5xzG3ofVT3H+FfJWs6NfaBq9xpmqQNBdW77XU/oR6g9Qa+364D4p/DaDxvo5uLJUi1m1Q+RJ080f882P8j2P1NdmGr8j5ZbCaPlCiprq1nsruW2u4nhnhcpJG4wVYcEEVDXrEhXrvwT+JH9g6ivh3WZsabdv/o8jniCU9v91v0PPrXkVLnHSoqQVSPKwPu2ivK/gt8Rf+Em0caJq02dUsU+R2PNxEOh+o6H8D616pXh1IOEuVlnxl49hEHxB12MYwt9L0GP4jXPV0vxE/5KRr3/AF/S/wDoRrmq92HwogKKKKoD6G/Zukz4d1qPJyt0jY9Mp/8AWr2ivGP2bodvhzWpsH57tFz9Ez/7NXs9eLif4rKWx8k/GP8A5KtrH++n/oC1w9dx8Y/+Srax/vp/6AtcRXrUv4cfQkSilorQBKKKKACiiigCzYX1xpmo299ZyGOe3kWSNx2YHIr7V0DVo9d8O2GqQY2XcCS4HYkcj8DkV8Q19W/BC8a7+FdgHOTBJLEOewYkfzrhxsbxUho9Br5v/aL/AOR4sP8ArwX/ANDavpCvm/8AaL/5Hiw/68F/9DaubCfxRvY8hopaK9ckSilpKACiiigApQcHI60lFAH1p8IfFj+K/AVu93J5l7ZH7NOx6tgfKx+q4/EGu6r54/Zx1Uw+I9V0tmO24thMo7ZRsfyf9K+h68TEQ5KjSKRl+J4PtXhPVoOf3lnKox/uGviY9a+6polngeJ/uyKVP0IxXw/qlq1jq93aOu1oJnjIPbDEf0rrwT0khM7/AOAtwIfilboes9rNGP8Avnd/7LX1JXx18NtSGk/EjRLp22J9qWNz7P8AKf8A0KvsWssYvfT8hoKKKK4hhXl/xX+FEPi22fVtEjWLWo1yyjgXQHY/7XofwPt6hRVwnKEuaIHwtcW81pcyW9zE0U0TFXjcYKkdQRUVfT3xX+FEPi22fVtEjWLWo1yyjgXQHY/7XofwPt8zXFvNaXMlvcxPFNExV43GCpHUEV7NKrGrG6IPRPhT8UZvBl6NO1Vnm0Wd/mHU27H+NR6eo/Hr1+obe4hu7aO4tZUmhlUOkiHKsD0INfCtep/CT4qyeFLpdH1yRpNHmb5HJybVj3H+ye4/H1zz4nD83vx3GmfTdFNjkSaJJYXV43UMrKchgehBp1eWUFFFcx478c6d4F0M3l6RLcyZW2tQcNK39FHc04xcnZAR+PvH2neBNFNxdETXsoItbQNgyH1Poo7mvk3Xtdv/ABJrU+qatO01zO2ST0UdlA7AdhUniTxHqPirXJtU1ebzZ5TwBwqL2VR2Aqpp2nXerajDY6dA9xczsEjjQZLGvZo0VSjd7kN3DTtOu9W1GGx06B7i5ncJHGgyWNfUvwx+GNp4G08XN2EuNZnX99MBkRD+4nt6nvR8MfhjaeBtPFzdhLjWZ0/fTAZEQ/uJ7ep7139cWIxHP7sdikgoooriGFFFBIVSTwByaAPkj4wXX2r4q6yc5Ecix/8AfKAVxkKGSeNFGSzAAetaXinUf7X8W6rf7twuLuR1PqpY4/TFS+DbA6p420azUbvNvYgw/wBncCf0Br34+7BeSIPs61iFvaQwjpGioMewxUtFFeAWFFFFABRRRQAUUUUAFFFFABWLqvg7w7rl79r1fR7S8uNoXzJY8nA6Ctqimm1sBzH/AArbwb/0Len/APfkUf8ACtvBv/Qt6f8A9+RXT0VXtJ9wOY/4Vt4N/wChb0//AL8ij/hW3g3/AKFvT/8AvyK6eij2k+4HMf8ACtvBv/Qt6f8A9+RXHeP/AIL6BeeHbq88OWQ0/ULaNpUSJjsmwMlSp6E9iK9YpkwBgkBGQVOQfpVRqzi7pgfCtJTn/wBY31pte6QFFFFAH1z8H/8AklOi/wDXN/8A0Y1drXFfB/8A5JTov/XN/wD0Y1drXg1f4kvUs5qT4deEJpXll8O2Du7FmYxDJJ6mm/8ACtvBv/Qt6f8A9+RXT0UuefcDmP8AhW3g3/oW9P8A+/Io/wCFbeDf+hb0/wD78iunoo9pPuBwHirwz4C8J+GbzWL7w3p5S3TKoIgDI54VR9TXy3ql9/aWpTXYtoLVZGysFum1Ix2AFe5ftH6y6Wuj6NGxCyM9zKM9cfKv82rwKvUwsXycz6ksKKKK6xHffB3wpYeLfHP2XV4jNaW1u1w8WSA5BUAEjtls/hX0Qvw08GIoUeG7DA9Ys14x+zn/AMjtqP8A14H/ANDWvo+vKxU5KpZMpFHSdF03QrM2uj2UNnAWLmOFcAse/wClXqKK4229xnP3vgTwtqV9LeX+hWVxczNuklkiBZj6mvKvjb8O9A0bwrHreh2MdjNFcLHKkXCOrZ5x6g46epr3WvNfj1/yS2f/AK+of51vQnL2kVcTPluiiivaJCruj/8AIcsf+viP/wBCFUqu6P8A8hyx/wCviP8A9CFD2A+4aKKK+dLCiivO/i18SE8FaQLPTmVtYvEPlDr5CdPMI/kPX6VcIOcuVAcF+0FZ+GVvoLq2uUTxAcCe3iGd8eOGf+6w7dyPpXiVTXV1Pe3UlzdyvNPKxZ5JGyzE9yahr26cOSCje5AUUUVoBe0bV7zQdYttT02UxXNtIHRv6H1B6EV9g+DfFtl4x8LwavZkKWG2eLPMMg+8p/mPYivjCus8C+PL7wTcXv2cGW2vbd45Id2AH2nY49wT+RNc2Io+0jdboaZj+Jbz+0PFWqXedwmu5XB9QWOKy6U80ldKVlYQUUVb0vTrjV9WtdPs0Lz3UqxRqPUnFAH038CNLbTvhjDM6lWvriSfnuOFH6LXpFUtF0uHRNDstMtv9VaQrEpx1wMZ/GrteBUlzTcizB1DwP4Z1a+kvdS0SzubmXG+WSPLNxiq3/CtvBv/AELen/8AfkV09FHPJdQOY/4Vt4N/6FvT/wDvyKP+FbeDf+hb0/8A78iunoo9pPuB5x4r+CvhjWdJnGj2Mel34UtDLASFLdgy5xg+3NfLc0TwTvDKpWSNirKexBwRX3XXxR4rUL4y1pVGAL+cAen7xq9DB1JSumyWZFFFFd4gr6h+AP8AyTFf+vyX/wBlr5er6h+AP/JMV/6/Jf8A2WuTGfwvmNHptY+r+EtA166S41nSbW9mRNivMm4hc5x+tbFFeSm1qijmP+FbeDf+hb0//vyKP+FbeDf+hb0//vyK6eiq9pPuBzH/AArbwb/0Len/APfkVj+I/g34S1rS5YbLTotMu9v7q4tgV2t2yvQiu/opqpNO6YHw7qum3OjatdadfJsuLWVopF9wcVTr03496Ytj8SWuI02i9to5Tjuwyp/9BFeZV7dOXPBSICiiirA9F+Bc5h+K1io6TQzIef8AYLfzUV9U18mfBj/krGkfWT/0W1fWdeVjP4i9CkFfJPxh0g6R8UNVXbiO6cXSe4cZP/j26vravDv2jdAL2ul6/Cn+rY2sxA7H5kz+TfnU4SXLUt3Bng0MrwTxzRHa8bBlPoRX2v4b1ePXvDGnapEcrdW6SH2JHI/A5FfElfRn7PXiYXvhy78PzvmWwfzYQT1jc8j8Gz/31XVjIXhzdhI9ioooryigooooAK8w+K/woh8W2z6tokaxa1GvzKOFugOx/wBr0P4H29Poq4TlCXNED4WuLea0uZLe6ieKaJirxuMFSOoIqKvp/wCK3woh8XWz6tosaxa1EvzKOBdAdj/teh/A+3zLcW81pcyW91E8U0TFXjcYKkdQRXs0qsasbohnrHwh+LB8Oyx6D4imZtLkbEE7HP2Unsf9g/p9K+kFZXUMhDKwyCDkEV8JV638NfjO/hbSZtK8QpNeWsMZazZDl0IHEZz/AAnse306c2Iw3N70NxpnuHjbxpp3gjQXv9QYPM2Vt7YHDTP6D29T2r5M8T+J9S8W65LqerzeZK/CIPuxL2VR2Ap/ivxXqXjDXZdT1aUszcRxA/LCnZVHp/Os7TtOu9W1GGx06B7i5nYJHGgyWNbUKCpK73BsNO0671bUYbHToHuLmdwkcaDJY19S/DH4Y2ngbThc3YS41mdf304GREP7ie3qe9L8MfhjaeBtOFzdhLjWZ0/fTgZEQ/uJ7ep7131ceIxHP7sdhpBRRRXEMKKKKACub+IOtjw94B1fUAwWRbdo4ucfO/yr+pz+FdJXhP7RXiYbdP8ADdvJ82ftVyAenZAf/Hj+VbUYc9RITPCK9K+A+kf2l8SorllzHp8Dzk+hI2j/ANCrzSvo/wDZ58P/AGHwpeazMmJNQm2Rkj/lmnH6sW/IV6uIly0mStz1+iiivELCiiigAooooAKKKKACiiigArzDx18Z4vBPiiTRn0R7wxxpJ5ougmdwzjG0/wA69Pr5a+PP/JVLn/r2h/8AQa6cNCM52kJnZ/8ADSsH/QsSf+Bw/wDjdH/DSsH/AELEn/gcP/jdeB0V6P1Wj2Juz3z/AIaVg/6FiT/wOH/xuu3+HHxNT4hSX6R6U1h9jVCSZ/M37s/7Ix0r5Mr3T9mv/j41/wD3IP5vWFfD04U3KKGme9U2X/Uv/umnU2X/AFL/AO6a8wo+FX/1jfWm05/9Y31ptfREBRRRQB9c/B//AJJTov8A1zf/ANGNXa1xXwf/AOSU6L/1zf8A9GNXa14NX+JL1LCiiiswCiiigD5s/aJlZvHdkjH5UsV2j0y7V5JXrP7Q/wDyP9r/ANeKf+hNXk1e5Q/hRIe4UUUVsB6/+zn/AMjtqP8A14H/ANDWvo+vnD9nP/kdtR/68D/6GtfR9eRi/wCKUtgooorkGFea/Hr/AJJbP/19Q/zr0qvNfj1/yS2f/r6h/nWtH+JH1Ez5booor3SQq7o//Icsf+viP/0IVSq7o/8AyHLH/r4j/wDQhQ9gPuGiiivnSynq+qW2i6Pd6lfPst7WJpXPsB0+p6V8Z+J/EF34o8R3mr37Ey3Dkhc8IvRVHsBgV7p+0P4jNn4fstBgfD30nnTAf8806D8W5/4DXzrXq4OnaPO+pLCiiiu0QUV7h8OfhxZeLvg1erOqx3tzdvJa3BHMZRQo/AncCPevGtT0260fU7jT9QhaG5t3MciN2IrONSMpOK3QFSiiitACiiigAr3X4BeBGMreLdTiIVQY7BWHU9Gk/oPxrjfhd8L7rxtqC3t+rwaLbv8AvZOhnI/gT+p7fWvqa1tYLK0itrSJYYIUCRxoMBVHAArgxVdJckdxpEtFFFeYUFFFFABRRRQAV8UeLf8AkdNb/wCwhP8A+jGr7Xr4o8W/8jprf/YQn/8ARjV6GC3kSzIooor0hBX1D8Af+SYr/wBfkv8A7LXy9X1D8Af+SYr/ANfkv/stcmM/hfMaPTaKKK8goKKKKACiiigD54/aQTHiXR5P71ow/J//AK9eMV7R+0i4PiPRk7raOfzf/wCtXi9e3h/4SIe4UUUVuB6J8DITL8WNPbGRHFM54/6ZsP5kV9VV83/s66cZ/GmoX5GVtbMpn0Z2GP0U19IV5GMd6pSCsTxj4ej8U+EdQ0iUDNxEfLJ/hkHKn8wK26K5U2ndDPha5t5bS6ltrhDHLC5R0I5Ug4Ird8CeKJfB/jGy1ZCxiRtlwg/jibhh/Ue4Fd58e/Bh0rxCniKyjxaaicT4HCTAdf8AgQGfqDXkNe7GUasL9yNj7ptrmG8tIrm1kWWGZBJG6nhlIyDUteH/AAG+IAmtx4S1WbEkYLWDufvL1Mf4dR7Z9K9wrxatN05crLCiiiswCiiigArzD4rfCiHxdbPq2ixrFrUS8qOFugOx/wBr0P4H29Poq4TlCXNED4Wubaa0uZLe6ieGaJijxuMMpHUEVFX0/wDFb4Uw+LrZ9W0WNYtaiXkDhboDsf8Aa9D+B9vm+20XUrvWl0i3s5n1BpfKFvtwwbuCO1ezSrRqRuRYi07TrvVtRhsdOge4uZ2CRxoMljX1N8MfhjaeBtOFzdhLjWZ1/fTgZEQ/uJ7ep70fDL4ZWngbThcXQS41mdP304GREP7ie3qe9d9XBiMRz+7HYpIKKKK4hhRRRQAUUUUAVNV1O10bSbrUr+QR21rGZJGPoP69q+MfE2vXPibxJe6veE+ZdSlgufuL0VfwGBXq3x4+IC310PCukzboLdw166nh5B0T6Dqff6V4pXrYSlyx5nuyWXNJ0y41nWLXTbJC891KsSD3Jxn6V9p6FpEGg6DZaVaf6q0hWJT64HJ/E5P414j+z74MMt1P4qvov3ceYLPcOrfxuPoOPxPpXvtc2Mqc0uVdBoKKKK4hhRRRQAUUUUAFFFFABRRRQAV8tfHn/kqlz/17Q/8AoNfUtfLXx5/5Kpc/9e0P/oNdmD/ifITPNqKKK9YkK90/Zr/4+Nf/ANyD+b14XXun7Nf/AB8a/wD7kH83rnxP8Jgtz3qmy/6l/wDdNOpsv+pf/dNeKWfCr/6xvrTac/8ArG+tNr6IgKKKKAPrn4P/APJKdF/65v8A+jGrta4r4P8A/JKdF/65v/6Mau1rwav8SXqWFFFFZgFFFFAHzV+0P/yP9r/14p/6E1eTV6z+0P8A8j/a/wDXin/oTV5NXuUP4USHuFFFFbAev/s5/wDI7aj/ANeB/wDQ1r6Pr5w/Zz/5HbUf+vA/+hrX0fXkYv8AilLYKKKK5BhXmvx6/wCSWz/9fUP869KrzX49f8ktn/6+of51rR/iR9RM+W6KKK90kKu6P/yHLH/r4j/9CFUqu6P/AMhyx/6+I/8A0IUPYD7hooor50s+UfjZq/8AavxQv0Vsx2Spapz02jLf+PE15/Wr4ovG1DxZqt25y015K+fq5rKr36ceWCRAUUUVYH1r8GYPI+E+jjGNwkf65kY1y/x0+H39r6YfE2lRZvbNMXSKOZYh/F9V/l9K6/4S/wDJK9D/AOuJ/wDQ2rsWVXQq6hlYYIIyCK8V1HCs5LuV0PhKiu8+LXgU+DPFjG0jI0y+zLakDhP70f4E/kRXB17EZKcVJEhVzSprO31e1m1O2N1ZpKpnhV9pdM8gHtxVOiqA+3dAn0y58P2U2giJdOeFTbrEuFVfTHb/ABrRr5x+BnxC/sbVB4a1WbFjeP8A6M7niGU9vYN/P6mvo6vDrU3TnZlIKKKKxGFFFFABRRRQAV8UeLf+R01v/sIT/wDoxq+16+KPFv8AyOmt/wDYQn/9GNXoYLeRLMiiiivSEFfUPwB/5Jiv/X5L/wCy18vV9Q/AH/kmK/8AX5L/AOy1yYz+F8xo9NoorjPGPxR0PwRqkNhq8V48ssQlUwRhhjJHcj0ryoxcnaJR2dFeV/8ADQvhH/n31P8A78r/APFUf8NC+Ef+ffU/+/K//FVp7Cr/ACiueqUV5X/w0L4R/wCffU/+/K//ABVVr/8AaJ8ORWbtp2n6hcXGPkSRVRc+53E4+gp/V6v8oXRxX7RN9HP44srVDl7ayAf2LMT/ACxXkdaOva3eeI9dutW1Jw1xdPvbHRfQD2AwKzq9inHkgokhRRWjoI0o61bnxC866erbphboGdwP4RkjGeme1W9EB9F/APw62k+Bn1K4TbNqkvmLkc+WvC/mdx/EV6lXk9v8ffBdpbRW9tZ6jFDEgSNFgQBVAwAPmrr/AAX8QNJ8dpdto0dygtSofz0C53ZxjBPpXi1oVG3OSKOpooornGZfiTw/Z+KPD13pGormG4TG7HKN1Vh7g4NfHXiTw/e+F/EF1pOpptmt3xuxw69mHsRzX2zXn3xX+HMfjfRBcWKqmsWakwMePNXqYz/Q9j9a68NW9nLlezE0fLFrdT2V3FdWkrQzwuHjkQ4KsOQRX1Z8L/iPb+ONGEV0yRaxbKBcQjjzB08xfY9x2P4V8pXFvNaXMlvcxNFNExR43GCpHBBFWdH1i+0HVoNS0q4a3uoG3I6/yPqD6V6FaiqsfMlH3BRXBfDf4o6d44s1trgpaaxGv722JwJPVk9R7dR+td7XjSjKDtIsKKKKkAooooAKzovD+lQ69LrUVhCupTRiN7kL8xUf56+wrRop3aAKKKKQBRRRQAUUUUAFeZ/Fz4mx+ENNbS9JlVtauU42nP2ZD/Gff0H4/Vfid8W7Pwhby6bpDx3WtMMbQcrbe7ep/wBn86+Y729udRvpry+nee4mcvJLIcsxPc13YfD8z557CbInd5ZGkkYu7ElmY5JPrW74L8KXnjLxPbaVZgqrHdPLjiKMfeY/09yKydPsLrVdQgsdPhee5uHCRxoMlia+s/hr4AtvAnh4RNtk1K5Ae7nHc9kH+yP1612V6ypR8yUjptJ0u10TSbbTdOj8q2toxHGvsO59z1q5RRXi7lhRRRQAUUUUAFFFFABRRRQAUUUUAFfLXx5/5Kpc/wDXtD/6DX1LXy18ef8Akqlz/wBe0P8A6DXZg/4nyEzzaiiivWJCvdP2a/8Aj41//cg/m9eF17p+zX/x8a//ALkH83rnxP8ACYLc96psv+pf/dNOpsv+pf8A3TXilnwq/wDrG+tNpz/6xvrTa+iICiiigD65+D//ACSnRf8Arm//AKMau1rivg//AMkp0X/rm/8A6Mau1rwav8SXqWFFFFZgFFFFAHzV+0P/AMj/AGv/AF4p/wChNXk1es/tD/8AI/2v/Xin/oTV5NXuUP4USHuFFFFbAev/ALOf/I7aj/14H/0Na+j6+cP2c/8AkdtR/wCvA/8Aoa19H15GL/ilLYKKKK5BhXmvx6/5JbP/ANfUP869KrzX49f8ktn/AOvqH+da0f4kfUTPluiiivdJCruj/wDIcsf+viP/ANCFUqu6P/yHLH/r4j/9CFD2A+4aZMxW3kYdQpI/Kn0hGVI9RXzpZ8M3rF9QuGbqZGJ/OoKuavGYdbvojwUuJFP4Map19EtiAooooA+ufg/J5vwp0U4xiN1/J2FdrXAfBGbzfhLpmTko8yHjp+9b+mK7+vBq/wASXqWcx8QfCEPjTwhc6a4UXKjzbWQ/wSgcfgeh9jXx9dW01ldy211G0c0LlHRhyrA4Ir7or54+P3gr7DqkXiiwixBeER3YUcLLjhv+BAfmPeuvCVbPkfUTPGKKKK9MkVWKsCpIIOQR2r6o+D3j8eMPDn2PUJQdWsFCy56zJ0WT+h9/rXytWz4V8SXnhPxJa6vp7fPC3zoTxIh+8p9iKwr0lVjbqCPtWis/Qdbs/Eeh2uq6bJvt7lA6+qnup9weDWhXitNOzLCiiikAUUUUAFfFHi3/AJHTW/8AsIT/APoxq+16+KPFv/I6a3/2EJ//AEY1ehgt5EsyKKKK9IQV9Q/AH/kmK/8AX5L/AOy18vV9Q/AH/kmK/wDX5L/7LXJjP4XzGj02vm/9ov8A5Hiw/wCvBf8A0Nq+kK+b/wBov/keLD/rwX/0Nq48J/FG9jyGiiivXJCiiigAooooAKKKKACvfP2bP+PXXv8Afh/k1eB175+zZ/x669/vw/yaubFfwmNbnuVFFFeMUFFFFAHlnxY+E8Xiq3k1jQY1j1mNcvGOBdAdj/teh79DXzTc201ncyW91E8M0TFHjdcMpHUEV901wPxF+FOmeOIWuoNtlq6rhLkL8snorjv9eo9+ld2HxPJ7s9hNHypbXM1ncx3FrK8M0bBkkjYhlI7givdfAHx6Rli07xtkNwqaii8H/roo/mPy71454i8Mat4V1RrDW7R7eUfdY8rIPVW6EVkV3zpwqx1J2PuizvLbULSO6sZ47iCQZSSJgysPqKmr4w8M+Ndf8I3Pm6HqEkKE5eBjujf6qePx617L4a/aI0+4VIfFOnyWkmObi1+dD9V6j8M151TCTjrHUq57VRWLo3jHw74gRW0jWLS4Zv8AlmJAH+m081tVyNNOzGFFFFIAooooAKKwNa8deGfDysdV1m1idQT5Svvc/wDAVya8t8T/ALRUKK8HhPTWkboLq84X6hByfxI+lawo1J7ILns+o6nZaRYyXmqXUVpbRjLSysFA/wA+leEfED48y3iy6b4L3wQn5Xv3GHYf7A/h+p5+leVeIfFeteKrz7Trt/LdMPuoThEHoqjgVj16FLCRjrLVk3HSSPLI0krs7sSWZjkk+pNT2Gn3eqX8Nlp8D3FzOwSOKMZLE1q+FfBus+MdSFnotq0gBHmzNxHEPVm/p1r6d8AfDTSfAlmGiAutTkXE1468/wC6o/hX9T3rWtXjSXmJIzfhb8LbfwRZC+1EJPrU6YdxysCn+Bf6nv8ASvRaKK8ic5TlzSLCiiioAKKKKACiiigAooooAKKKKACiiigAr5a+PP8AyVS5/wCvaH/0GvqWq81hZ3EnmT2kEr9Nzxgn8zW1Gr7KXNYTPhqivuL+ydO/58LX/vyv+FH9k6d/z4Wv/flf8K7Prq/lFY+Ha9z/AGbP+PnX/wDcg/m9e3/2Tp3/AD4Wv/flf8KlgtLa13fZreKHd18tAufyrOrilUg42HYmpsv+pf8A3TTqK4Rnwm/+sb60lfcX9lad/wA+Fr/35X/Cj+ydO/58LX/vyv8AhXpfXV/KTY+HaK+4v7J07/nwtf8Avyv+FH9k6d/z4Wv/AH5X/Cj66v5Qscr8H/8AklOi/wDXN/8A0Y1drTYoo4YxHCixovRUGAPwp1efKXNJsoKKKKkAooooA+av2h/+R/tf+vFP/QmryevuaaxtLl99xawysBjdJGGOPxqP+ydO/wCfC1/78r/hXfTxahBRtsKx8O0V9xf2Tp3/AD4Wv/flf8KP7J07/nwtf+/K/wCFX9dX8orHz3+zn/yO2o/9eB/9DWvo+oILK1tnLW1tDCxGCY4wpI/Cp6461T2k+awwooorEYV5r8ev+SWz/wDX1D/OvSqZNBFcR+XcRJKmc7XUMPyNXCXJJS7AfCtFfcX9k6d/z4Wv/flf8KP7J07/AJ8LX/vyv+Fd/wBdX8pNj4dq5o//ACHLH/r4j/8AQhX2t/ZOnf8APha/9+V/wpRpWnggiwtgRyCIV4/Sj66v5QsWqKKK80o+OfiRpx0v4ka5bFdo+1tIo/2X+Yfo1cxX3LLp9nPIZJ7SCRz1Z4gSfxIpn9k6d/z4Wv8A35X/AAr0Y4yySaJsfDtFfcX9k6d/z4Wv/flf8KP7J07/AJ8LX/vyv+FP66v5Qsedfs/XIm+GjRZ5t72VMemQrf8As1eoVHBbwWyFbaGOFSckRqFBP4VJXBUlzycu5QVm+ItDtfEnh680i+XMN1EUzjlD2Ye4OD+FaVFSm07oD4f1rSbnQtau9Lvk2z2spjcY647j2PWqVfcsun2c8hkntIJHPVniBJ/Eimf2Tp3/AD4Wv/flf8K9BY3TWJNj4dor7i/snTv+fC1/78r/AIUf2Tp3/Pha/wDflf8ACn9dX8oWPnf4GePf7D1v/hHtTmxYag/7hmPEUx4H4N0+uPevpOqo0rT1YEWNsCDkEQrx+lWq461SNSXMlYaCiiisRhRRRQAV8U+Lf+R01v8A7CE//oxq+1qqtpdg7FnsbZmY5JMKkk/lXRQreybdriaufDlFfcX9k6d/z4Wv/flf8KP7J07/AJ8LX/vyv+FdX11fyisfDtfUHwB/5Jiv/X5L/wCy16F/ZOnf8+Fr/wB+V/wqeGCK3j2W8SRJnO1FCj8hWNbEqrHlsNIkr5v/AGi/+R4sP+vBf/Q2r6QqCeytblw1zbQysBgGSMMQPxrGjU9nPmsB8M0V9xf2Tp3/AD4Wv/flf8KP7J07/nwtf+/K/wCFdn11fyisfDtFfcX9k6d/z4Wv/flf8KP7J07/AJ8LX/vyv+FH11fyhY+HaK+4v7J07/nwtf8Avyv+FH9k6d/z4Wv/AH5X/Cj66v5QsfDtFfcX9k6d/wA+Fr/35X/Cj+ydO/58LX/vyv8AhR9dX8oWPh2ve/2bP+PXXv8Afh/k1ez/ANk6d/z4Wv8A35X/AAqaC1t7XP2aCKHd18tAufyrKrilUg42HYloooriGFFFFABRRRQBm674e0vxLpr2Gt2cd1A3QOOUPqp6g+4rwbxn8AdS05pLvwlKdQtuT9lkIEyD0B6N+h+tfRdFbU606fwisfDF5ZXWn3T219by286HDRyoVYfgagr7Y13wtoniW38nXNNguwPus6/Ov0Ycj8DXlev/ALOmn3BaTw5qstoe0N0vmL9NwwR+tehDGQfxaCsfPqsysCpII6EGt3TPHPijSNq6fr1/Ci9E89mUf8BORXRav8EvGulljHp6X8YPD2kobP8AwE4P6VyN74c1rTWIv9Jvbfb1MkDKB+OK6VKE+qYj7M0G4luvDum3Fw5eWW1id3I+8xQEn86v1meGuPCmk54/0KH/ANAFadeC9yzB8cX9zpngTWb6wlMNzb2ckkUigEqwHB5r5K1Lxl4k1gMupa5f3CN1Rp22/wDfIOK+rviMC3w18QBQSTYy4AH+zXyZY+Fte1Nwtho99cFuhS3Yj88V6ODUeVtksyiSxyTk+9JXoukfA3xpqhUz2cOnRt1a6lAI/wCArk/pXo3h/wDZ30ezZZfEOozag46xQr5SfnyT+ldMsRTj1FY+f9O0y+1e8S00y0mu7h/uxwoWJ/KvY/Bf7P1zcGO88ZTfZ4uosYGy7f7zdF+gyfpXt+jeH9J8PWv2fRdPt7KPuIkwW+p6n8a0a4qmMlLSGhVilpWj6foenx2OkWkVpbRj5Y4lwPqfU+5q7RRXDe+4wooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACiiigAooooAKKKKACkKhhhgCPQilooAAMDA4FFFFAAQGBBGQeoNAAAwBgUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAFFFFABRRRQAUUUUAf//Z)

Software Engineering Department

Braude College

Capstone Project Phase B

**Pseudepigrapha Detection using Neural Imposters**

**23-1-R-21**

**Students**

Asaf Davidovitch 314863945

Eliyahu Arnson 330319294

**Supervisors**

Dr. Renata Avros

Prof. Zeev Volkovich

Contents

[1. Abstract 3](#_Toc125309439)

[2. Introduction 3](#_Toc125309440)

[3. Background and Related Work 4](#_Toc125309441)

[3.1. Machine Learning 4](#_Toc125309442)

[3.2. Classification 4](#_Toc125309443)

[3.3. Clustering 4](#_Toc125309444)

[3.4. Preprocessing 5](#_Toc125309445)

[3.5. Tokenizer 5](#_Toc125309446)

[3.6. Training 5](#_Toc125309447)

[3.7. Word Embedding 6](#_Toc125309448)

[3.8. Neural Network 12](#_Toc125309449)

[3.9. Dynamic Time Warping 13](#_Toc125309450)

[3.10. Isolated Forest 13](#_Toc125309451)

[4. Expected Achievements 14](#_Toc125309452)

[5. Research Process 14](#_Toc125309453)

[5.1. The Process 14](#_Toc125309454)

[Part A 14](#_Toc125309455)

[Part B 14](#_Toc125309456)

[5.2. Pseudepigrapha Detection Algorithm 15](#_Toc125309457)

[5.2.1. Overview 15](#_Toc125309458)

[5.2.2. Definitions 15](#_Toc125309459)

[5.2.3. Algorithm 15](#_Toc125309460)

[6. Graphic User Interface 17](#_Toc125309461)

[7. Evaluation plan 19](#_Toc125309462)

[8. Summary 19](#_Toc125309463)

[References 20](#_Toc125309464)

[Links 21](#_Toc125309465)

# Abstract

This project presents an implementation of a new method for pseudepigraphic detection. The algorithm in use is based on a previously implemented method called “Imposters Method” that uses a collection of imposters. The imposters are utilized to train the neural network to distinguish between two different literature creations. In this project the focus will be on improving the algorithm by replacing a key component, Word Embedding, with a more modern technique. In this project book extensive research was done for different embedding techniques, mainly on two progressive models BERT and GPT-2. A Transformer embedding procedure will be implemented, more specifically BERT model developed by Google, in order to achieve more accurate results in pseudepigraphic detection. The BERT embedding method offers rich contextual understanding of the literature under test. A broad background section gives a detailed understanding of the basics of neural networks, machine learning, and various embedding methods that are employed in the algorithm. At the end of this project book a detailed explanation of the algorithm can be found where each step of the algorithm is explained. A visual representation of the algorithm interface and its flow is located after the algorithm.

# Introduction

Pseudepigrapha are works with incorrect attribution, texts wit which the author attributed to the work is not the actual author. Authorship veriﬁcation is a daily problem that occurs in many different ways. For example, the importance of this matter can be seen in cases where documents that have great financial or legal importance and the author is unattributed. A well-known topic revolving around this subject is the Shakespeare authorship question. This matter discusses the idea that another author, other than Shakespeare, was responsible for the creation of works attributed to Shakespeare. According to some extreme theories, claims have been made that many of Shakespeare’s works, if not all, were in fact not written by William Shakespeare.

Studies have been done that show a correlation between the style of writing and its specific author. In a study done by Mendenhall, he proposed to “analyze a composition by forming what may be called a ‘word spectrum,’ or ‘characteristic curve,’ which shall be a graphic representation of an arrangement of words according to their length and to the relative frequency of their occurrence." [12]

The writing style of an author can be from words used in a similar fashion, statements or phrases commonly used or even the flow of the text. This idea has led to a method of detecting outliers of works attributed to an author. This means that by using the styling of authors, an understanding can be formed in the field of authorship verification. This information is the key to understanding the direction in which the problem can be solved. The question is what methods can be applied to harness this “styling correlation” to classify correctly works done by the same author and works that would be considered pseudepigrapha in an effective manner. The next section explains basic concepts relating to Artificial Neural Networks and Word Embedding (Word2Vec, Transformers).

# Background and Related Work

The methods that currently exist in the field of pseudepigrapha identification:

1. The Chi-Square Method / Distribution: “The Chi-Square Method is a statistical method for evaluating the relationship between expected values and observed values” [1]
2. Type / Token Ratio: “This method is commonly used for small sized documents. The number of tokens, say ‗n‘ in a text is said to be equal to the number of words in that text. But even if the text has ‗n‘ words, all the words are not unique and many of them are repeated.”[1]
3. CUSUM Technique: “Every author has his/her own and unique writing style. The author tends to use some of the words frequently. These words are helpful in determining the author. This technique starts by measuring the length of the sentences in terms of number of words. The proper nouns exercised in the sentences are treated as single symbols. The average sentence length for a particular document is calculated and then each sentence is compared with the average length and is marked with a ‗+‘ or a ‗-‘ according to its length.” [1]

Pseudepigrapha Identification using neural networks is not something new. The tools that neural networks offer are very fitting for solving the issue at hand. The study about author authentication was written by Moshe Koppel and Yaron Winter. The article showed how to determine if two documents were written by the same author by testing their similarity. Later on, Moshe Koppel and Shachar Seidman wrote an article about automating identifying Pseudepigraphic texts using the “Impostors method”. In this matter there is an input of literature work that needs to be classified as written by a certain author. In order to achieve this, a trained neural network can be utilized to recognize the writing styles of specific authors. In order to use neural networks, the literature articles need to be transferred to numerical representations. In this project word embedding is used to take literature articles and “translate” the literature to an appropriate format for neural networks.

### Machine Learning

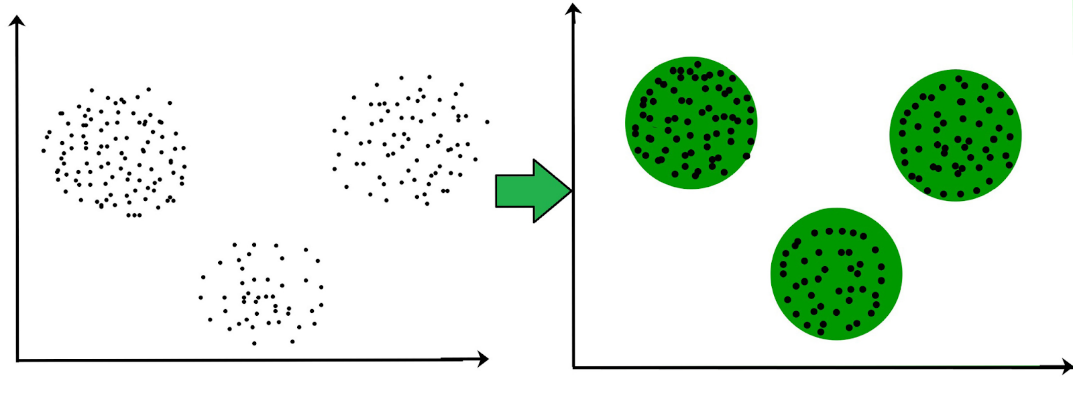
Deep learning is one subject in the broader field of machine learning. Deep learning is also a form of neural network with three or more layers. This form of neural network is closely modeled after a human brain, and it can process and learn from a vast amount of data.

### Classification

Classification is defined as the method of recognition and grouping of objects into predefined categories in order to distinguish between different data points. Classification is a natural process that humans perform in daily tasks. As humans, we classify between different types of vehicles, for example cars, trucks, and vans.

### Clustering

The process of establishing similarities between data and grouping the data into “clusters”. This allows for identifying patterns within a certain data set. The main idea is that objects with similar features (defined by the neural network) will be clustered together.



**Figure 1: Clustering Reference [9]**

### Preprocessing

The first step when performing preprocessing is to remove all the “noise” from the text.   
Because the dataset contains literature texts, the noise removal will focus on anything that is not related to textual human language, for example special characters, use of parentheses, use of square brackets and so on. In addition, all characters are transferred to lower case characters and stop words are removed due to lack of meaning. The result of this stage is “clean” text containing only meaningful words.

### Tokenizer

The Tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements. The token occurrences in a document can be operated directly as a vector representing that document. This immediately turns an unstructured string (text document) into a numerical data structure suitable for machine learning. They can also be utilized directly by a computer to trigger useful actions and responses. Or they might be used in a machine learning pipeline as features that trigger more complex decisions or behavior. A GPT-2\BERT trained tokenizer will be employed for the tokenization process.

### Training

Training is the processing of developing the neural connections between nodes in the network. The connection between the nodes consists of weights which define the inner personality of the network. The weights of the network are adjusted using the feedback loop called Gradient Backward Propagation. The backward propagation gives the network the feedback it needs to decide if the result of the network fits the expected results. If needed the weights are changed and as a result the output of the neural network is changed.

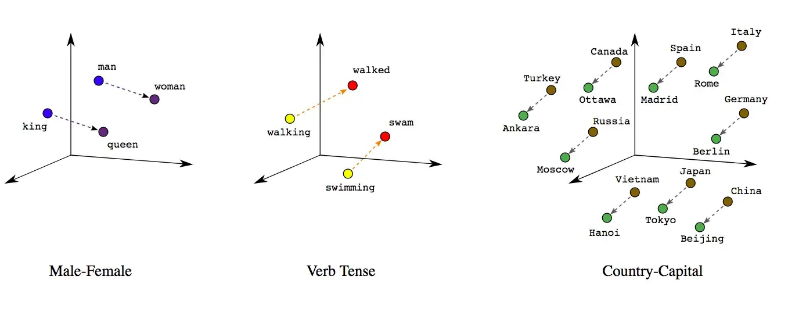
“One of the main challenges of deep learning methods is the choice of an appropriate training strategy” [7]. Post Training is the process of fine tuning an existing model to a more specific task. There is no one way for Post Training, every task will require a different Post Training method. In general Post Training is the process of adding or removing layers from the network and then , using the new network, run data on a small number of epochs (iterations) in order to fine tune the weights in the network.

### Word Embedding

Although a neural network is similar to a human brain in its architecture, the way in which a neural network understands the input is different. Neural Networks use float vectors that represent words, images, sounds and so on. The output of a network is also a vector representation of the real-world result. In order to communicate with a neural network, it is needed to translate the input, in its current form, to a float value vector. This process is called Embedding and it is an important part of using a neural network.

Word Embedding main stages:

1. Data Preprocessing - Matching the type of data to the vector representations. For example, the number of words in the dictionary is the dimension of the vector.
2. Tokenization - assigning a number for each word in the corpus.
3. Training Embedding Layer - training the layers to understand the vectors

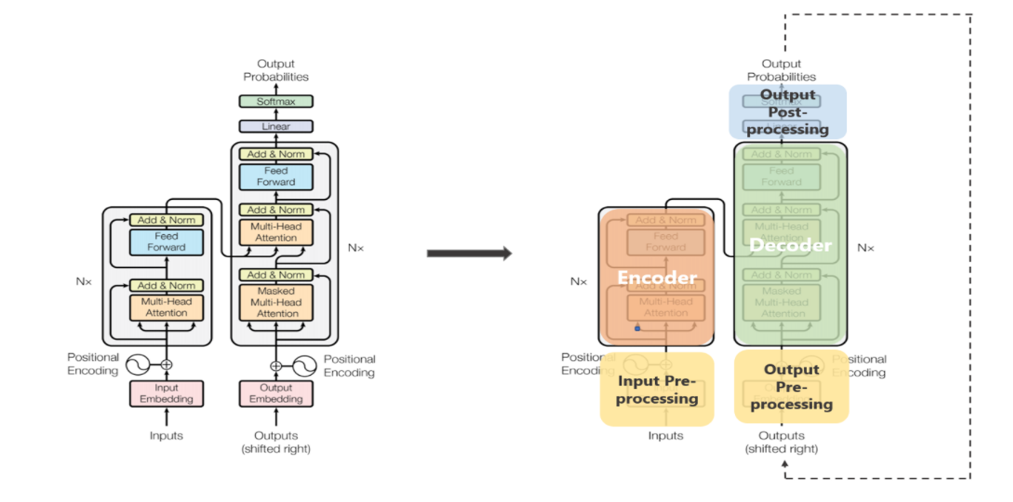
There are many methods of embedding that can be applied in order to achieve a vector representation of the input. In this project a focus will be on two methods Word2Vec and Transformers.

**Figure 2: Word Embedding Reference [11]**

#### Word2Vec

This method uses a neural network of two layers in order to create a vector representation of the input data. The idea behind this method is that inputs with similarities will have vectors with similar values. For example, a vector representing the word “London” will be close to a vector representing the word “Tokyo” because both words represent capitals of countries.

#### Transformers

Transformers are built to process input data in a sequential form. This design is why transformers are applied for natural language processing (translating from one language to another) as well as text summarization. The transformer uses a self-attention mechanism to understand the context of the input, in order to output a result reflecting a relation between the words in the input. As well, transformers process the entire input at once which can reduce the training time. Processing the entire input at once allows for the transformer to instill context into the vector representation. The Transformer consists of two main components which are called the Encoder and the Decoder. Both of the component’s architecture are similar with small differences and will be explained in depth later in the article. In the vanilla architecture of a standard Transformer the Encoder and Decoder are combined to create one single model. The transformer architecture:

**Figure 3: Transformer Reference [2]**

##### Encoder

The overall job of the encoder is to receive an input (a sequence of words) and to produce a vector representation of each word. This is done using the N = 6 identical layers, in which each of those layers has 2 sub-layers. This maps all input seq into an abstract continuous representation that holds the learned information for that entire sequence.

##### Decoder

The responsibility of the decoder is to continually receive input of features and output the next predicted word. The decoder works in an interactive fashion, predicting the output and feeding it into the next iterations input.

##### BERT

BERT known by the full name, Bidirectional Encoder Representations from Transformer, is a natural language processing model. The model was researched by Google AI Language and showcases drastic improvements in the areas of Natural Language Inference. An important component of BERT is the innovative design in the bidirectionally training of the Transformer. In the paper released by Google AI [4] a correlation between bidirectional learning and a deeper understanding of the contextual texts was found.

The model uses the attention mechanism to create a deep connection between words or sub words. As stated in the name, the model uses only the Encoder mechanism from the transformer. This allows BERT to be a language generating model. The encoder model in its original fashion processes the entire input at once, using the multi-head self-attention, which gives the model a deeper understanding of the surrounding words for each word.

Graphical user interface, text, application

Description automatically generated

**Figure 4: BERT Model Reference [10]**

###### BERT Architecture

As mentioned before, BERT consists of several layers of Encoders, where the output of one Encoder layer is the input for the next Encoder layer.  
The Encoder layer architecture:

* Input Embedding: Can be viewed as a lookup table for representing the input words as vectors.
* Positional Encoding: Information is added relating to the positions of the input embeddings. In short, for every odd time stamp a position vector is created with a cosine function and for every even time stamp a positional vector is created using a sin function. These positional vectors are then added to their corresponding input embedding vectors. This gives the network information on the positions of each vector.
* Chart

  Description automatically generatedSelf-Attention Mechanism: This allows the model to associate each individual word in the input to other words in the input. It can also learn that input structured into a certain pattern can be in the form of a question and so on. This mechanism uses query, key and value vectors. The query and key vectors are sent through a dot product multiplication to create a score matrix. The score matrix is a matrix of values representing the importance of each word to the other words. To avoid exploding effects of multiplied values the score matrix values are divided by the square root of

**Figure 5: Self-Attention Reference [8]**

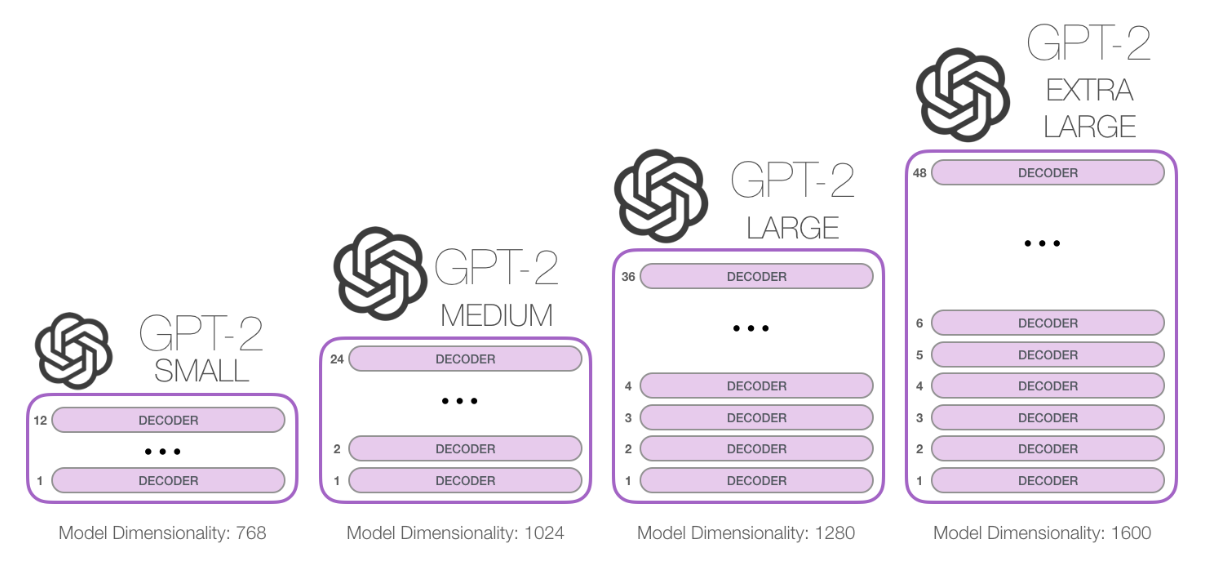
the dimensions of the queries and keys. A SoftMax is done on the scores in the score matrix resulting in values between 0 and 1. SoftMax amplifies higher scores and suppresses lower scores to raise confidence. The result of applying the SoftMax of the score matrix produces the attention weights. The attention weights are multiplied by the value vectors to receive the output vectors. This whole process is done with the N layers (each set is called a Head). The result of the layers, also known as the heads, are concatenated together. This allows for different heads to learn different information resulting in better final outputs.

* Residual Connections: These residual connections go around the attention mechanism and the feed forward mechanism. This helps the network to train by allowing gradients to flow through the network directly.
* Layer Normalizations: Stabilize the networks which results in reducing the training time necessary.
* Position Wise Feed Forward Network: Further process the attention output potentially giving it a richer representation.

In conclusion, the model processes the received input to a vector representation with attention information between the words in the text. The representation is continuous and without breaks. In this format a neural network can receive this representation as an input. By stacking encoder layers the model builds a richer understanding of the representations. This understanding improves the predictive ability of the network significantly.

##### GPT-2

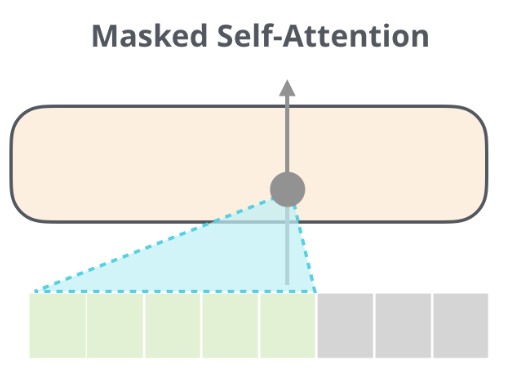
Generative Pre-trained Transformer 2 (GPT-2) is an open-source artificial intelligence created by OpenAI in February 2019. The model is a Natural Language Processing (NLP) model, that is able to write coherent essays that exceed what current language models are able to produce. It is able to translate text, answer questions, summarize passages, and generates text output on a level that can become repetitive or nonsensical when generating long passages. The model is based on the Transformer model, but it contains only decoder layers (the number of layers derives from the model size - small, medium large and Extra-large). GPT-2 comes in several sizes - according to the user’s needs. The different sizes allow for flexibility in accomplishing tasks with varying complexity.



**Figure 6: GPT-2 Model’s Reference [8]**

###### GPT-2 Architecture

As mentioned before, GPT-2 consists of several layers of Decoders, where the output of one Decoder layer is the input for the next Decoder layer. The decoder layers make the GPT-2 model fitting for language generation. The decoder has a similar inner structure to the encoder. The main difference is the use and amount of the inner components. The decoder has two Multi-Headed Attention modules each with a separate job. An important distinction between the encoder and decoder Self Attention modules is the Masked Self-Attention aspect. The Masked Self-Attention “restricts” the module’s contextual understanding of the input by “masking” over the values representing the tokens of the future words. This is located in the upper right triangle of the Scores Matrix (Shown in the following figure).



**Figure 7: Masked Self Attention Reference [8]**

###### GPT-2 Stages

The decoder receives the first starting key <s> as the input, known as a token. The token is sent through the Masked Self-Attention process. The main goal of the self-attention process is for the model to give each word a contextual score against the other words (tokens) sent before it. Each word is processed in this manner and the result of this model is a running score representing the contextual connection of each word to each word. The following figure shows the score for all the previous words against the word “it”.

Table

Description automatically generated

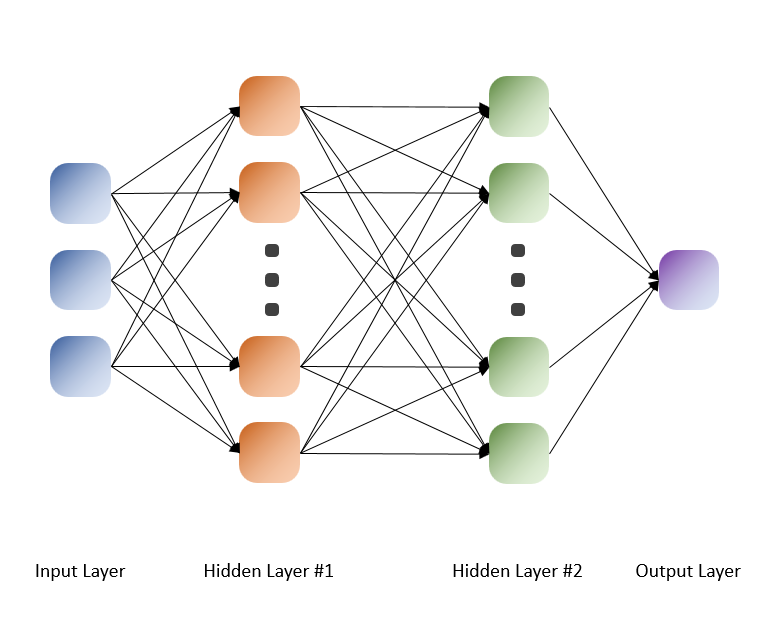
**Figure 8: Token Scores Reference [8]**

It can be shown that 50% of the attention was given to the word “robot” for the word “it”. After being processed by the self-attention layer the vector is passed through two layers of neural network called Feed Forward Neural Networks. The Feed Forward Neural Networks processes the vector representation to prepare the vector to be an adequate input for the next decoder layer. After the top block (the last decoder) an output vector is received, the result of its own self-attention and neural network processing). The vector result is then multiplied by the Token Embedding matrix to calculate the output token probabilities. The variable top\_k can be defined to show how many of the tokens to relate to in the result. The output is then fed into the input of the module and the decoder layers start over the stages. These stages are repeated until an <end of sequence> token is received.

#### Conclusion of Word Embedding

In conclusion the BERT model and the GPT-2 model have a lot in common, but the main goal of GPT-2 is to predict the next word, while BERT's main goal is to output a vector that contains as much contextual and rich information as possible about the input. The goal of the project is to research and implement an improved method of Word Embedding. After researching the methods GPT-2 and BERT, it has been concluded that the BERT model will give us a much deeper understanding of the literature writing style. This feature will give us better results in pseudepigrapha detection.

### Neural Network

Artificial Neural Networks are a set of nodes which are similar to the neurons in a human brain. Each connection, edge, between the nodes represents the synapses. These edges can pass signals, or in this case values, from one layer of nodes to another in the network. Each node processes the data received to it and passes it on to the next layer. The processing done in each node is what will lead to the required result. Weights can be placed on edges and nodes to modify the results as it is being passed through the network. The purpose of the weights is to increase or decrease the importance of the edges between nodes. Nodes can also have a threshold in which they can allow or stop the signal from continuing on.

**Figure 9: Neural Network Layout**

The network is divided into layers of nodes. These layers allow for different transformations to be done to the result passing through the network. Each layer can have a different purpose or continue from the previous layer. Neural Networks are built of at least 3 layers. Input layer, the first layer in which the input data passes through. Hidden layers, there can be many hidden layers utilized to process the signal. Output layer, this is the final layer and the result of the network. Similarly, to a brain, the Neural Network needs to be trained in order to accomplish a certain task. Training is the process of fine tuning the weights within the network to achieve the required output. When training a neural network, the most appropriate data set must be chosen in order to achieve the specific result. For example, using a neural network to identify dogs in a photo would require training the network with many pictures of dogs.

#### Convolution Bidirectional LSTM

After the stage of word embedding is completed, the output is a rich contextual representation of literature. These embeddings can be inputted into the neural network. The main goal of the neural network is to extract relevant features from the input in order to give more information for the Bidirectional-LSTM stage.

##### Convolution layer

A convolution is a filter to a given input. Repeatedly using the same filter on the same input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

##### RNN

RNN (recurrent neural network) is an artificial neural network that uses data over different time stamps or sequential data as input. The output is a vector that contains information about the entire input sequence. Recurrent neural networks are distinguished by their “memory” feature as they take information from prior inputs to influence the current input and output. While traditional deep neural networks assume that inputs and outputs are independent of each other, the output of recurrent neural networks depend on the prior elements within the sequence.

##### LSTM

LSTM (Long Short-Term Memory) are a type of RNN neural networks, capable of learning long-term connection. LSTMs are specifically designed to avoid the vanishing exploding gradient problem that RNN networks deal with by using cells that store data from previous cells. This feature allows the network to remember long term information.

##### Bidirectional LSTM

BiLSTM (Bidirectional LSTM) is a sequence processing model. BiLSTM consists of two layers of the LSTM unit. The first unit is responsible for taking the input in a forward direction, and the second unit is tasked with taking the input in reverse. BiLSTMs increase the amount of information that the network can get from the input sequence.

### Dynamic Time Warping

Dynamic Time Warping in short, DTW, is employed to compare temporal sequences that are not synced up perfectly. This method allows for capturing the optimal matching of sequences under comparison. DTW is utilized to calculate distances between two sequences. The process includes dividing the series into equal points, calculate the Euclidian distance between the first point in the first series against all the points in the second series and saving the minimum, repeating this for all the points and finally adding up all the minimum distances that were calculated. This value represents the similarity of the two series.

### Isolated Forest

An algorithm utilized for anomaly detection. The anomalies are calculated using isolation, distance of a data point from the all the other data and works off of a decision tree. The algorithm performs Anomaly Detection by locating outliers, data points which may differ drastically from the majority of other data points. This subject is important because real world datasets can be large and complicated and locating anomalies may not be trivial.

# Expected Achievements

It is expected to achieve better results when the algorithm uses the BERT model for Word Embeddings mainly because of the rich contextual understanding of the text. The results of the improved algorithm (BERT Embedding) will be compared with the results of the previous algorithm (Word2Vec) using the accuracy of the two algorithms.

# Research Process

This project discusses the problem of pseudepigrapha detection using neural imposters. The problem was suggested solutions, but most of these solutions use outdated methods. An updated solution to the problem is presented using new and upcoming methods that have only recently been published.

An improvement can be made in the embedding method of the previously implemented solutions to the problem to achieve better results. In this project a focus is placed on the utilization of an embedding model called Transformers that was published by Google Brain, “Attention Is All You Need” [2]. Another point of research done was on comparing different NLP models, specifically focused on BERT and GPT-2 (the second version of GPT). Research was done on which of the models will produce better results in the algorithm, or maybe using a combination of them both.

## The Process

Part A

The starting point of this project was understanding the individual components of the algorithm. After further research on each component, a deeper foundation was developed understanding the algorithm as a whole and its use. In order to study the subject of Transformers the article [2] was exploited as well as independent research on the subject from vast sources that elaborate on the Transformers. Research on the subjects of pseudepigrapha detection was done using the article [3].

Part B

In this part of the project, the understanding and research done will be leveraged to properly implement a Transformer method of embedding. The goal is to achieve a working algorithm for pseudepigrapha detection that relies on a Transformer embedding method. The second goal is to see an improvement in the pseudepigrapha detection from the methods implemented until today.

## Pseudepigrapha Detection Algorithm

### Overview

In this section the algorithm stages are laid out and each component is explained. To begin a Preprocessing of the literature is done, next Tokenization, embedding method in us, next the Convolutional Bidirectional LSTM neural network, calculate the DTW matrices, running Isolate Forest algorithm over the DTW matrices and finally grouping the results.

### Definitions

The algorithm uses 2 sets of data: Deriving from the first set of data are two imposters. The first imposter consists of literature works written in English that are known to be attributed to a specific other than Shakespeare, . The second imposter consists of literature works written in English that are known to be attributed to a specific author that is different from the first author and is not Shakespeare, . The second set, ,will be defined for the testing data, which consists of many literatures works from multiple authors, with at least one work that is known to be written by the tested author.

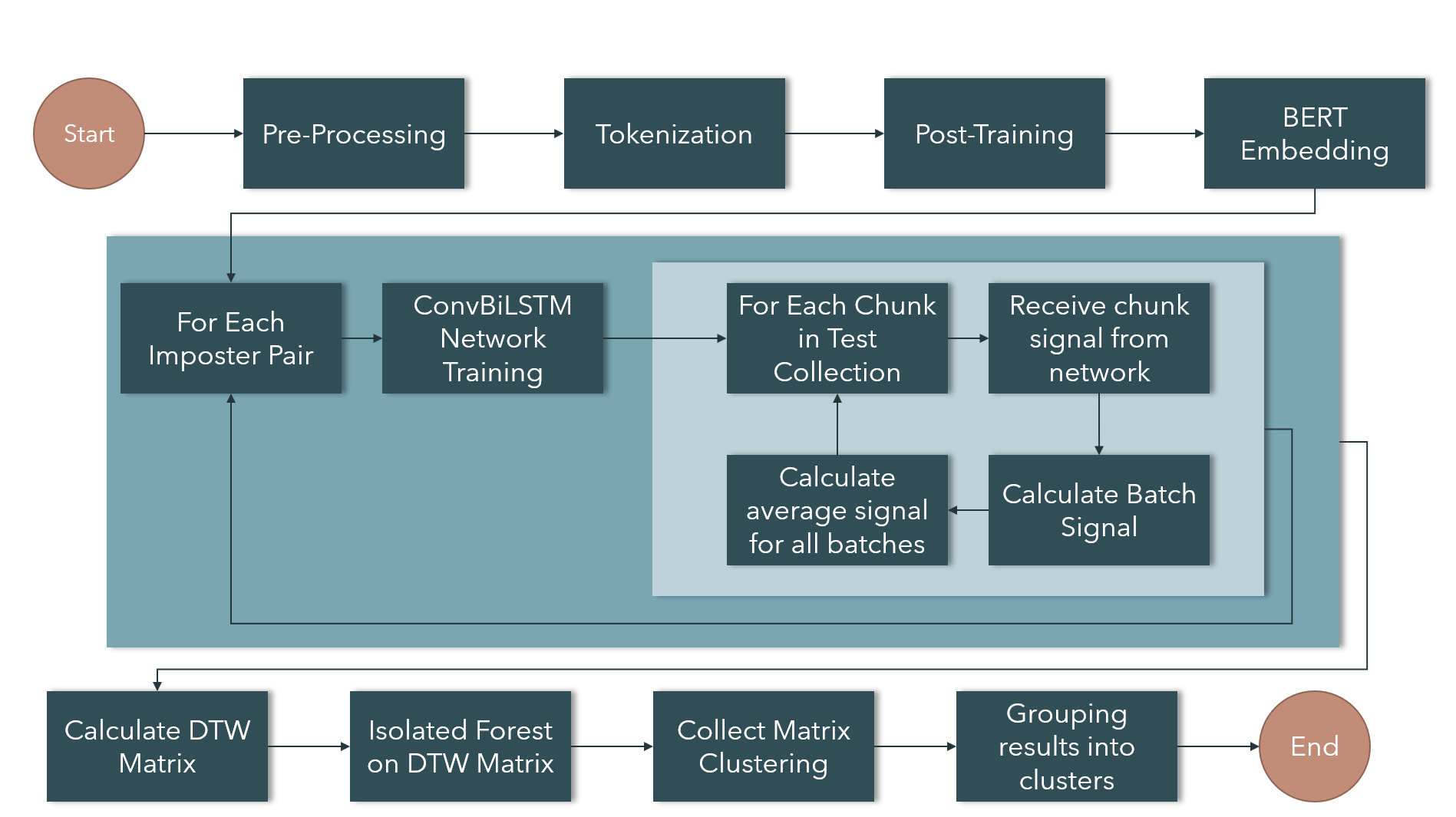
The neural network uses a list of parameters. The parameters are Number of kernels, Sizes of the convolution kernels, stride size, num of feature maps, num of neurons, data batch size, pooling size, learning rate, momentum, decay, dropout rate, number of epochs, activation function, loss, cross validation, output size, optimizer, and metric.

### Algorithm

The first step is to perform preprocessing on in order to remove the unnecessary characters that do not attribute value to the task at hand. Next, translating the data into a format more suitable for the task will be done. The tool utilized here is a Tokenizer. The tokenizer divides the data into chunks of a defined length. The results of the tokenized data are saved in a set, . Now that the data is divided into a more usable format, the data is fed into the transformer embedding model, BERT, to receive the contextual vector representation of each word. The vector pool is saved as a set representing all the words in the set, . Next, a post-Training procedure is performed on the pre-trained BERT algorithm by adding an extra layer and then training the entire model on the Imposters sets for 3 epochs. The additional layer is to fine-tune the BERT output to better suit the goal.

Now the following loop is performed:

1. For each randomly chosen imposters pair:
   1. Train the ConvBiLSTM network using the imposters pair.
   2. For each Document in the Test collection ():
      1. Feed the network with and get a signal for each chunk.
      2. Calculate the batches signals.
      3. Calculate the average signal of the batches.
2. Calculate the matrix of the pairwise DTW distances.
3. Run Isolated Forest algorithm 1000 times over the DTW matrix.
4. Collect the matrix clustering.
5. Grouping the results into clusters.

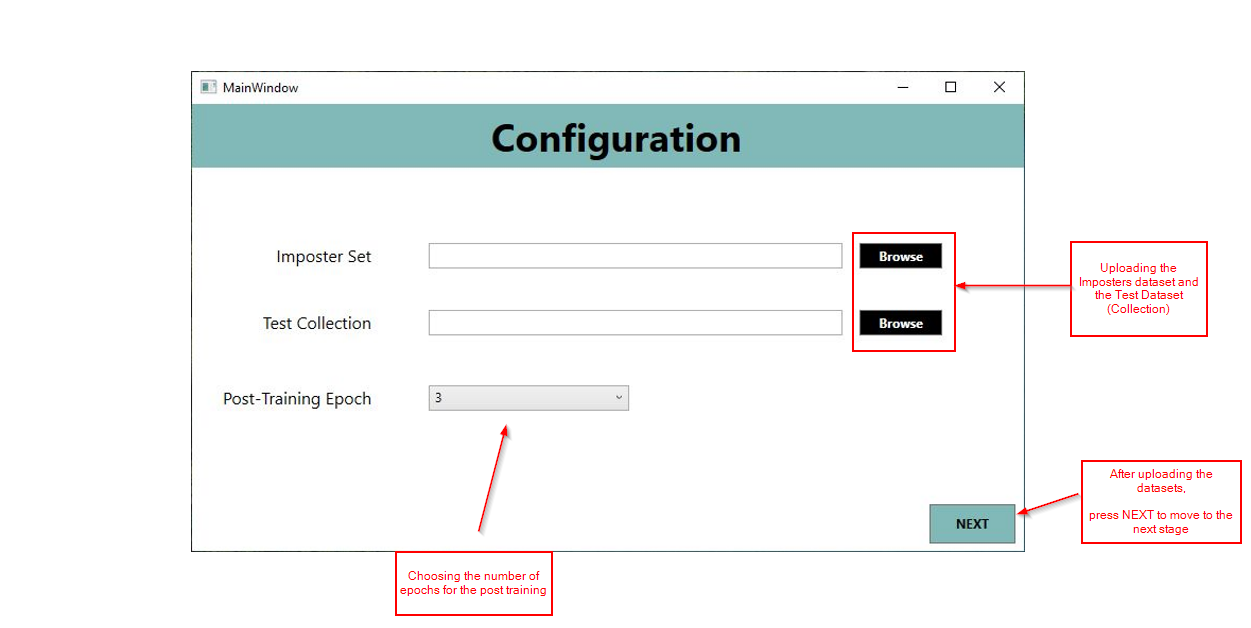
According to the clusters data it can be noticed that documents with similar writing style are found to be clustered together. It can be inferred from this that documents in the same cluster are written by the same author. The fact that a document of the tested author was placed with the testing data allows us to indicate that the other documents in the same cluster were written by the tested author.

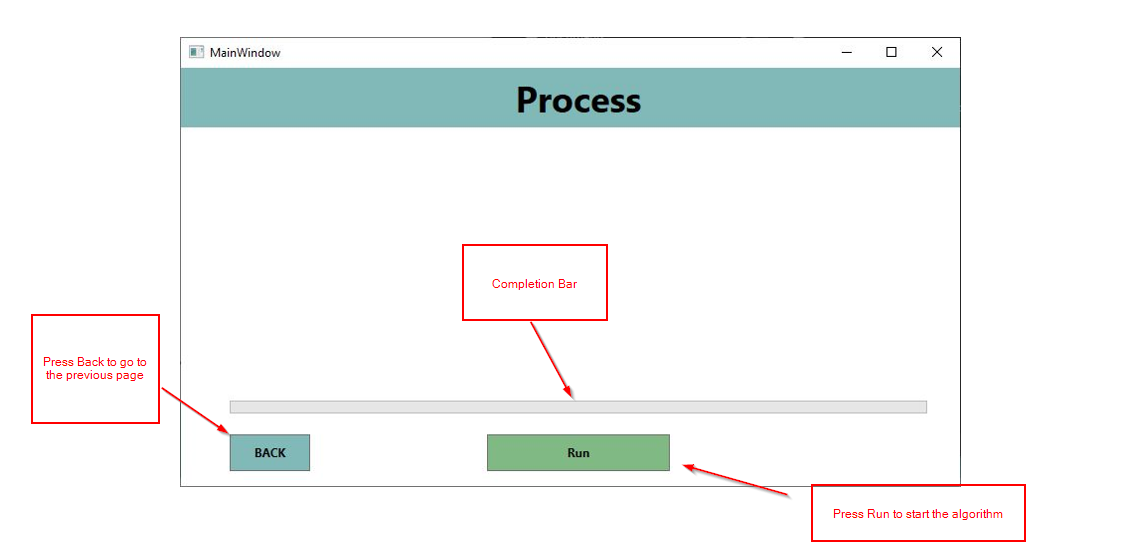
# User Documentation

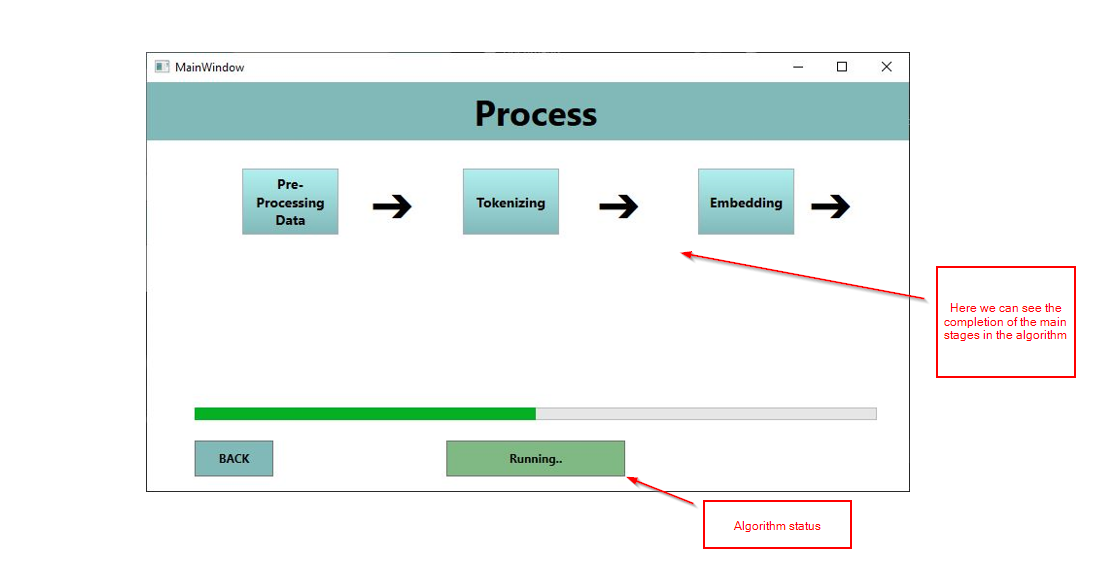
# User Guide

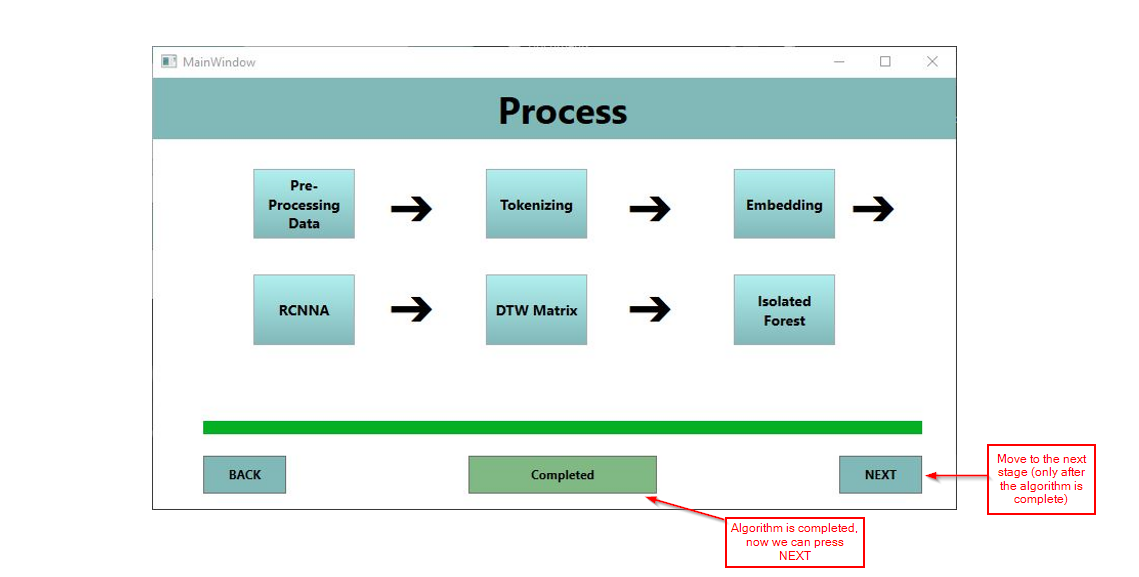
The application is intended to visualize the described algorithm. In addition, the application allows the user to analyze the resulting clusters. The application is built using Visual Studio 2019 as a WPF application. The application displays for the user the final texts that are suspected to be a pseudepigrapha in a very easy to understand way.

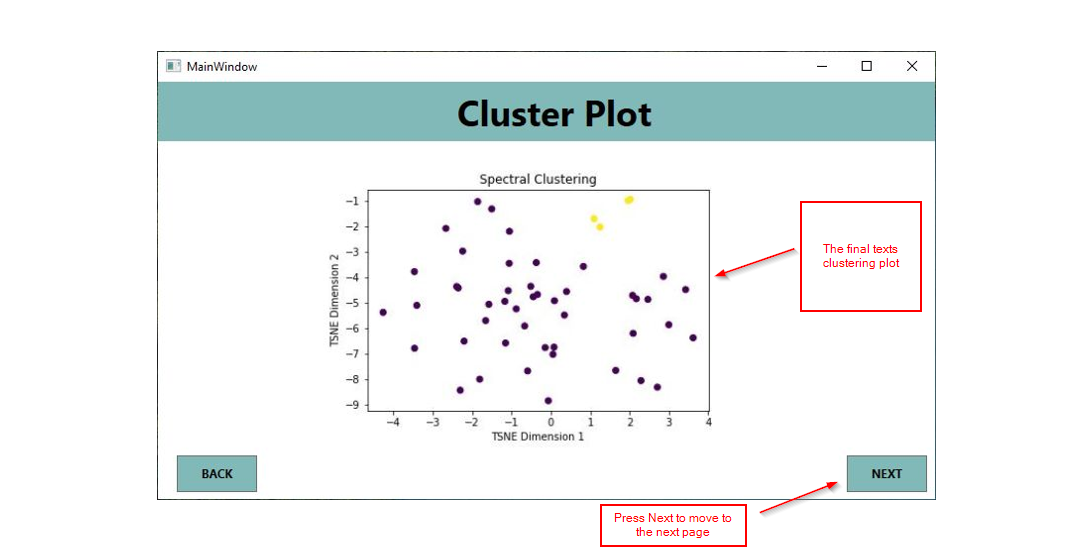
# User GUI

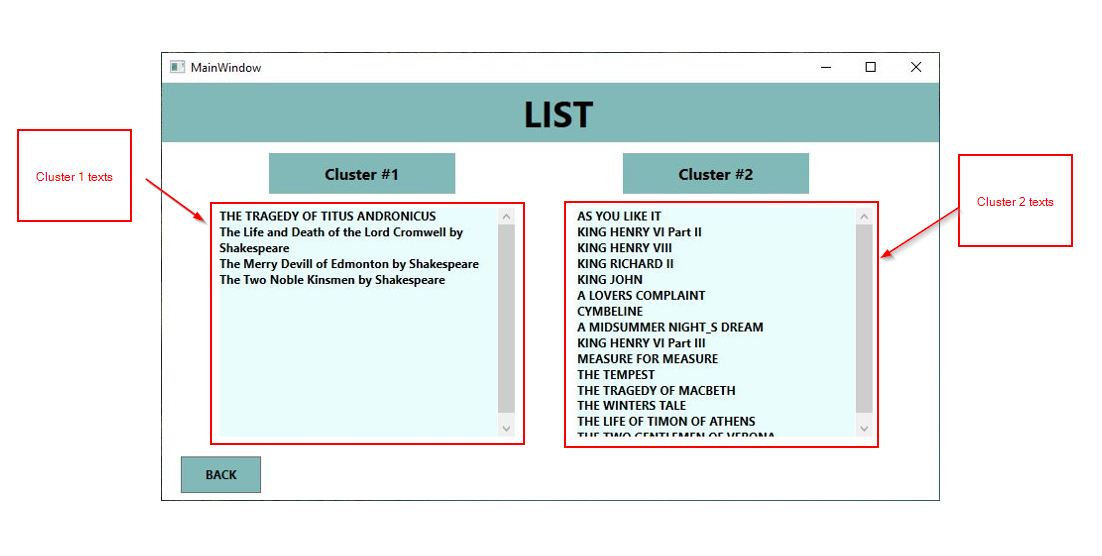












# Evaluation plan

In part B of the project, tests of the results of the updated algorithm will be done, using BERT transformer, against the results of the previously implemented algorithm, using Word2Vec embedding. The process of comparing the algorithms will use the literature classification accuracy of the algorithms. The purpose of the tests is to validate algorithm correctness using different methods. The algorithms will be compared using the same input data and comparable parameters in order to receive accurate results.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | Interface / Module | Testing Function | Expected Result |
| 1 | Application Backend | Preprocessing | Successfully preprocess the data |
| 2 | Application Backend | Tokenization | Successfully tokenize the processed data |
| 3 | Application Backend | BERT post training | Post train the model using the selected portion of data |
| 4 | Application Backend | Calculate the signals matrices | Successfully calculate each literature matrix |
| 5 | Application Backend | Calculate DTW matrix | Successfully calculate the matrix |
| 6 | Application Backend | Isolate forest | Successfully run the algorithm |
| 7 | Application Backend | Prepare the results | 2 Separate clusters |
| 9 | Application Frontend | Configuration page | Successfully display all elements, uploading folders, receiving input |
| 10 | Application Frontend | Process page | Successfully display all elements, dynamic completion bars |
| 11 | Application Frontend | Graph Results page | Successfully display all elements and graphs |
| 12 | Application Frontend | Clusters Results page | Successfully display all elements and both lists of the results clusters |
| 13 | Application Frontend | UX | Intuitive use |
| 14 | Application Frontend | Pages switching | Fast, each button sends the user to the correct page |
| 16 | Application Frontend | Hamburger button (Menu button) | Correctly show all page’s names (as buttons) |

# Summary

The topic of pseudepigrapha Detection is introduced and the importance of the issue is during our daily life is elaborated. A deep explanation into the world of neural networks and its uses are described. An improvement is presented for an existing algorithm by changing the algorithm’s embedding stage from Word2Vec method to the Bert model. The improvement of the embedding method using a newly introduced technique is expected to show promising results. By focusing on a foundational component of the algorithm more accurate results can be achieved.

## Development Process

## Research for libraries

We started by searching for existing libraries for using the BERT model and we found two main libraries: Hugging Face and TensorFlow.

The libraries gave a very similar results, so we chose to use the TensorFlow library.

## Using the BERT model

We had to understand step by step exactly how to prepare the BERT model output results for the ConvBiLSTM network.

At the beginning we succeeded in running the full algorithm, but the ConvBiLSTM didn’t learn how to classify the dataset correctly during the training process in each iteration of the loop (in each iteration of the loop we choose two different imposters, and we train the ConvBiLSTM using those imposters).

We searched for solutions and found that an improvement can be made by slicing each imposter’s text into smaller chunks and embed those chunks using the BERT model.

To be more specific we preprocessed each of the texts to have an embedding output matrix with the dimension of (50, 768). This step was performed in order to match the input dimension from the original algorithm.

After we added text slicing into chunks to the preprocessing stage, the ConvBiLSTM performed the training successfully.

## Comparing results of BERT and Word2Vec

We created a testing plan in order to compare the models:

1. Comparing the ConvBiLSTM loss and accuracy results at each step.
2. Comparing the ConvBiLSTM loss and accuracy results in the first 10 iterations and in the last 10 iterations.
3. Comparing the final clusters results.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

## Trying to Fine Tune Word2Vec

After comparing the results of the algorithm that uses the BERT model for word embedding with the algorithm that uses Word2Vec for word embedding we noticed that the results were very close.

Importing the BERT model and the implementation is less efficient than using the Word2Vec model, considering that we want to create an app that runs the algorithm, and we want our app to be efficient in space and time, we decided to try to fine tune the Word2Vec model instead of using the BERT model.

We researched how to fine tune the Word2Vec model, and we found that we can perform fine tuning by adding the imposters texts (after the preprocessing) to the model’s vocabulary.

## Analyzing the fine-tuned Word2Vec

Once we successfully fine-tuned the Word2Vec we wanted to evaluate if the received results were an improvement to the original Word2Vec results.

The exact same tests from paragraph 9.3 were used to compare the results of the fine-tuned Word2Vec model with the results of the original Word2Vec, the tests are:

1. Comparing the ConvBiLSTM loss and accuracy results at each step.
2. Comparing the ConvBiLSTM loss and accuracy results in the first 10 iterations and in the last 10 iterations.
3. Comparing the final clusters results.

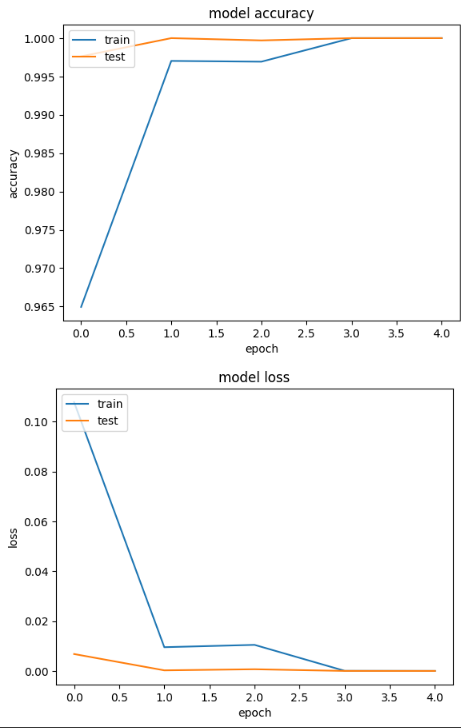
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

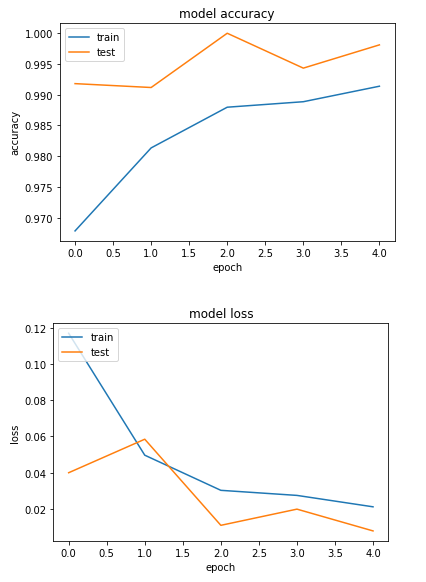
**11.3 Fine-Tuned Word2vec vs Original Word2Vec**

Here is an example of the ConvBiLSTM results when using the Fine-Tuned Word2Vec and the results when using the original Word2Vec. The following comparison shows the improvement between the fine-tuned model and the original model:

**Fine-Tuned Word2Vec**  **Original Word2Vec**

****





# References

[1]: Deshmane, N. *et al.* (2015) *Study of different methods for author identification*, *International Journal of Engineering Research & Technology*. IJERT-International Journal of Engineering Research & Technology. Available at: https://www.ijert.org/study-of-different-methods-for-author-identification (Accessed: January 4, 2023).

[2]: Vaswani, A. *et al.* (2017) *Attention is all you need*, *arXiv.org*. Available at: https://arxiv.org/abs/1706.03762 (Accessed: January 4, 2023).

[3]: Koppel, M. and Seidman, S. (2013) *Automatically identifying pseudepigraphic texts*, *ACL Anthology*. Available at: https://aclanthology.org/D13-1151/ (Accessed: January 4, 2023).

[4]: Devlin, J. *et al.* (2019) *Bert: Pre-training of deep bidirectional Transformers for language understanding*, *arXiv.org*. Available at: https://arxiv.org/abs/1810.04805 (Accessed: January 4, 2023).

[5]: Alammar, J. (2019) *The illustrated GPT-2 (Visualizing Transformer language models)*, *The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time.* Available at: https://jalammar.github.io/illustrated-gpt2/ (Accessed: January 4, 2023).

[6]: Xu, H. *et al.* (2019) *Bert post-training for review reading comprehension and aspect-based sentiment analysis*, *arXiv.org*. Available at: https://arxiv.org/abs/1904.02232 (Accessed: January 4, 2023).

[7]: Moreau, T. and Audiffren, J. (2018) *Post-training for Deep Learning*, *OpenReview*. Thomas Moreau. Available at: https://openreview.net/forum?id=H1O0KGC6b (Accessed: January 4, 2023).

[8]: Alammar, J. (n.d.). *The illustrated GPT-2 (Visualizing Transformer language models)*. The Illustrated GPT-2 (Visualizing Transformer Language Models) – Jay Alammar – Visualizing machine learning one concept at a time. Retrieved January 22, 2023, from <https://jalammar.github.io/illustrated-gpt2/>

[9]: GeeksforGeeks. (2023, January 11). *Clustering in machine learning*. GeeksforGeeks. Retrieved January 22, 2023, from <https://www.geeksforgeeks.org/clustering-in-machine-learning/>

[10]: GeeksforGeeks. (2022, June 20). *Explanation of Bert Model - NLP*. GeeksforGeeks. Retrieved January 22, 2023, from https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[11]: Google. (n.d.). *Embeddings: Translating to a lower-dimensional space  |  machine learning  |  google developers*. Google. Retrieved January 22, 2023, from <https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space>

[12]: *The characteristic curves of composition : Mendenhall, T. C. : Free download, Borrow, and streaming* (1887) *Internet Archive*. Science. Available at: https://archive.org/details/jstor-1764604 (Accessed: January 22, 2023).

# Links

A link to the git repository for the project is stated below:

* [Final Project Git Repo Link](https://github.com/asafdav95/Final-Project)