УПРАВЛЕНИЕ ДОСТУПОМ В ПРИЛОЖЕНИЯХ НА БАЗЕ ФРЕЙМВОРКА RUBY ON RAILS

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Медунецкий М.А., Хоронеко М.П.

Стержанов М.В. – к.т.н., доцент

В настоящее время на многих сайтах присутствует система управления доступом и разделения прав на выполнение действий. Наибольшую популярность в веб-программировании получила система на основе ролей, которая и будет описана в этой работе.

Управление доступом на основе ролей — развитие политики избирательного управления доступом, при этом права доступа субъектов системы на объекты группируются с учётом специфики их применения, образуя роли[1]. Например, существует несколько типов пользователей сайта со своими правами. Гости могут только смотреть информацию, зарегестрированные пользователи оставлять коментарии, модераторы удалять их, а администраторы изменять содержимое веб-ресурса.

Рассмотрим пользователя сайта, который представлен моделью User. Модель пользователя должна содержать следующие поля: логин или email, пароль (обычно в целях безопасности хранится хэш пароля) и роль. Зачастую роль представлена перечислением и в базу данных сохраняется номер элемента в этом перечислении. Ruby On Rails предоставляет удобный метод взаимодействия с базой данных через класс ActiveRecord. Модель наследуется от этого класса и получает удобные методы для выполнения CRUD (создание, чтение, изменение и удаление) операций.

Для аутентификации и работы с ролями удобно использовать gem Devise [2], который содержит методы для проведения этих действий.

Аутентифика́ция — процедура проверки подлинности. Эта процедура выполняется следующим образом. Если посетитель сайта не вошёл в систему, то отправляет запрос, который содержит логин и пароль. Сервер ищет такого пользователя в базе данных и в случае успеха генерирует и возвращает токен доступа. В дальнейшем этот токен отправляется в каждом запросе, обычно заголовком. Сервер получает запрос и сопостовляет полученный токен с пользователем, которого можно получить методом current_user, если пользователь не идентифицирован, то метод возвращает nil. Devise содержит и другие методы для проверки аутентификации посетителя.

Авториза́ция — предоставление определённому лицу или группе лиц прав на выполнение определённых действий. Для установки ограничений на доступ описываются методы, которые возвращают булевое значение. Затем они прикрепляются к контроллеру, либо к его методам. Это позволяет запретить пользователям получение или изменение информации, чья группа не уполномочена на такие действия.

Таким образом, фреймворк Ruby on Rails предоставляет все необходимые инструменты для создания и поддержки приложения с управлением доступом на основе ролей, а gem Devise значительно упрощает эту работу.

Список использованных источников:

1.FerraioloD. F., KuhnD. R. (October 1992). "Role Based Access Control". 15th National Computer Security Conference: 554—563.

2. https://github.com/plataformatec/devise - Flexible authentication solution for Rails on github

РЕКУРРЕНТНАЯ НЕЙРОННАЯ СЕТЬ LSTM ДЛЯ ПРОГНОЗИРОВАНИЯ РЕЗУЛЬТАТОВ СОРЕВНОВАНИЙ ПО КИБЕРСПОРТУ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Минчук С.Г.

Жвакина А. В. – канд. техн. наук, доцент

Возрастающая популярность киберспорта, большое количество соревнований по всему миру, в том числе международных, рост призового фонда, а также признание нескольких киберспортивных дисциплин настоящим видом спорта во многих странах заставляют относиться к данным мероприятиям не как обычным развлечениям.

Прогнозирование направлено наопределение тенденций развития конкретного объекта или события на основе предыдущих данных, иными словами, анализа его состояния в прошлом и настоящем [7]. Результат игры базируется не только на данных в определенный момент, а также данных, собранных на протяжении всей игры до этого момента. Рекуррентные нейронные сети− это сети, основанные на использовании предыдущих состояний для вычисления текущего, поэтому необходимо рассмотреть возможность

использования для прогнозирования рекуррентной сети LSTM, которая способна к обучению долговременным зависимостям.

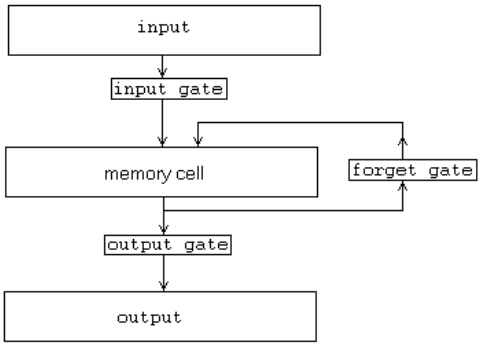

Рис 1. Схема нейронной сети LSTM

LSTM состоит из следующих частей [2] (рис.1): вход (*input*) нейронной сети; выход (*output*) нейронной сети; внутреннее состояние нейронной сети или запоминающая ячейка (*memorycell*); фильтр очистки памяти или фильтр забывания (*forgetgate*); входной фильтр или фильтр обновления памяти (*inputgate*); выходной фильтр или фильтр выдачи результата (*outputgate*).

Рассмотрим подробнее структуру рекуррентной сети LSTM [2]. Центральным понятием здесь является запоминающая ячейка, которая, наряду с состоянием сети g, вычисляется на каждом шаге, используя текущее входное значение $x^{(t)}$ и значение ячейки на предыдущем шаге $c^{(t-1)}$, где t это шаг обучения.

Входной фильтр определяет, насколько значение запоминающей ячейки на текущем шаге должно влиять на результат. Значения входного фильтра лежит в интервале [0, 1], что обеспечивается областью значений функции активации:

$$g_i = \varphi(W_i x + R_i y + P_i C + b_i) \quad (1)$$

где W_i– веса входа сети для входного фильтра,
x – вход сети на текущем шаге,
y – выход сети на предыдущем шаге,
R_i – веса выхода сети для входного фильтра,
C – состояние памяти сети,
P_i – веса состояния памяти для входного фильтра,
b_i– сдвиг значений входного фильтра,
φ – функция активации значения входного фильтра.

Фильтр очистки памяти удаляет часть информации запоминающей ячейки, вычисленной на предыдущем шаге:

$$g_f = \varphi(W_f x + R_f y + P_f C + b_f) \quad (2)$$

где W_i– веса входа сети для фильтра очистки памяти,
x – вход сети на текущем шаге,
y – выход сети на предыдущем шаге,
R_i – веса выхода сети для фильтра очистки памяти,
C – состояние памяти сети,
P_i – веса состояния памяти для фильтра очистки памяти,
b_i– сдвиг значений фильтра очистки памяти,
φ – функция активации для значения фильтра очистки памяти.

На основе данных входного фильтра и фильтра очистки памяти (1) и (2), поступающих на текущем шаге, вычисляется состояние запоминающей ячейки:

$$C(t+1) = Zg_i + C(t)g_f \quad (3)$$

где $C(t)$ – состояние запоминающей ячейки в текущий момент времени,
Z – изменение памяти,
g_i – значение входного фильтра,
g_f– значение фильтра очистки памяти.

Изменение памяти Z вычисляется по следующей формуле:

$$Z = \tanh(W_m x + R_m y + b_m) \quad (4)$$

где x– вход сети на текущем шаге,
W_m – веса входа сети для запоминающей ячейки,
y – выход сети на предыдущем шаге,
R_m– веса выхода сети для запоминающей ячейки,
b_m – сдвиг значения запоминающей ячейки,
$\tanh$ – функция активации для запоминающей ячейки.

Далее рассчитывается значение выходного фильтра. Выходной фильтр аналогичен двум предыдущим и имеет вид:

$$g_o = \varphi(W_o x + R_o y + P_o C + b_o) \quad (5)$$

где W_o– веса входа сети для выходного фильтра,

x – вход сети на текущем шаге,
y – выход сети на предыдущем шаге,
R_o – веса выхода сети для выходного фильтра,
C – состояние памяти сети,
P_o – веса состояния памяти для выходного фильтра,
b_o– сдвиг значений выходного фильтра,
φ – функция активации для значения выходного фильтра.

Итоговое значение сети определяется выходным фильтром (5) и нелинейной трансформацией над состоянием запоминающей ячейки (3):

$$y = tanh(C)g_o \quad (6)$$

гдеC– состояние памяти сети, g_o– значение выходного фильтра, $tanh$ – функция активации для выходного значения сети [2].

Повышенная невосприимчивость к продолжительности разрывов во времени является преимуществом LSTM над обычными рекуррентными нейронными сетями, марковскими моделями и другими методами обучения для последовательностей во многих областях применения.

Суть прогнозирования игры заключается в том, что на основе данных нейронной сети, собранных до текущего момента времени, она определит вероятность победы одной из команд.

На вход в нейронную сеть поступают следующие параметры: количество добытых ресурсов каждой из команд; номер команды, вероятность выигрыша которой мы хотим узнать; текущие предметы каждого из персонажей; количество убийств и использованных вспомогательных предметов, разрушенных зданий противника и оставшихся собственных зданий; опыт каждого из персонажей; распределение способностей у персонажей.

Все время одной партии игры t разбито на промежутки $t = (t_0, t_1, \dots, t_n), t > 0$. Набор игровых параметров представлен вектором $v(t) = (v_0^t, v_1^t, \dots, v_m^t)$, в конкретный момент времени игры t. Номер команды, вероятность выигрыша которой мы хотим узнать (может принимать значения 0 или 1) обозначен c. Входным набором параметров в нейронную сеть будет номер команды c и матрица:

$$X = \begin{pmatrix} v(t_0) \\ v(t_1) \\ \dots \\ v(t_n) \end{pmatrix} = \begin{pmatrix} v_0^{t_0}, v_1^{t_0}, \dots, v_m^{t_0} \\ v_0^{t_1}, v_1^{t_1}, \dots, v_m^{t_1} \\ \dots \\ v_0^{t_p}, v_1^{t_p}, \dots, v_m^{t_p} \end{pmatrix}, p < n \quad (10)$$

Выходным параметром нейронной сети является вероятность $y[0, 1]$ выигрыша команды c.

После обучения нейронной сети на большом объеме выборки, ее можно использовать не только для прогнозирования результата игры, но также в качестве помощника. Dota2 – это командная игра и в начале происходит стадия выбора героев. Данную нейронную сеть можно применить для выбора наилучшего героя в зависимости от уже выбранных. Также ее можно будет применить для выбора наилучшего предмета для героя в зависимости от стадии игры.

Нейронные сети будут эффективны при обучении аналитиков, участников соревнований, разработке более сложных и интеллектуальных стратегий, а также прогнозировании результатов соревнований.

Списокиспользованныхисточников:

1. Result Prediction by Mining Replays in Dota2. [Электронный ресурс] – Режим доступа: https://www.diva-portal.org/smash /get/diva2:829556/FULLTEXT01.pdf. – Дата доступа: 14.03.2018.

2. Рекуррентная нейронная сеть. [Электронный ресурс] – Режим доступа: http://mechanoid.kiev.ua/neural-net-lstm.html. – Дата доступа: 14.03.2018.

ПОСТРОЕНИЕ КРИПТОСТОЙКОГО ГЕНЕРАТОРА ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Молчанов И.В.

Калугина М.А.–канд. физ.-мат. наук, доцент

Источники настоящих случайных чисел найти крайне трудно. Ими могут быть, например, физические шумы, такие, как детекторы событий ионизирующей радиации, дробовой шум в резисторе, или космическое излучение. Однако реальное применение таких чисел порождает ряд проблем (время- и трудозатраты при установке и настройке, невозможность воспроизведения ранее сгенерированной последовательности и другие). Поэтому на практике чаще используются генераторы псевдослучайных чисел.

Генератор псевдослучайных чисел (ГПСЧ) — это алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и подчиняются заданному распределению (обычно равномерному). Во многих задачах современной информатики качество получаемых результатов напрямую зависит от качества используемых ГПСЧ.