React-Recommender

Jan 23, 2020

Contents

CHAPTER 1

Quick Start

Install the react-recommender library.

```
npm install --save react-recommender
```

This is a full example. It optimizes the displayed message ("Hello World" or "Hello You") based on which option better achieves the objective: making the user click the button.

```
import Recommender, { Recommend, Option, withObjective } from 'react-recommender';

const Objective = withObjective(({onAchieved, ...props}) => <button onClick={(evt)=>␣
↪onAchieved("OptimizeClicks")}>Click Me</button>);

<Recommender accountId="mail@react-architect.com">
    <div>
        <Recommend
            mode="egreedy"
            epsilon={0.1}
            objectiveId="OptimizeClicks"
            options={[
              <Option id="helloWorld">
                <div>Hello World</div>
              </Option>,
              <Option id="helloYou">
                <div>Hello You</div>
              </Option>
            ]}>{
                ({loading, recommendation, error, renderOption}) => {
                    return (loading && <div>Loading</div>) ||
                        (recommendation ? renderOption(recommendation) : <div>Error</
↪div>)
                }
            }</Recommend>

        <Objective />
```

(continues on next page)

(continued from previous page)

```
    </div>
</Recommender>
```

CHAPTER 2

Resources

- How To Build A Self-Improving React App–It is easier than you might think

2.1 Getting Started

2.1.1 Install Library

You can install react-recommender easily.

```
npm install --save react-recommender
```

2.1.2 Integrate React-Recommender

react-recommender provides a higher-order-component as the default export. Integrate it at a high level in your app.

```
import Recommender from 'react-recommender';

ReactDOM.render(
  <Recommender accountId="mail@react-architect.com">
    <MyApp/>
  </Recommender>,
  document.getElementById('root')
);
```

The `<Recommender/>`-component takes a single property: `accountId`. You can use any arbitrary `string` here. But if you want to see the statistics of how well your React app performs, you'll need to specify an email address. It ensures that no one else gets access to it.

2.1.3 Define An Objective

The most important aspect of a self-improving algorithm is an objective. The objective of your React app is up to you. Typical objectives of web-apps are clicks on a button, subscriptions, or transactions. Whatever fits the purpose of your app.

While you can choose any arbitrary objective, it must be measurable by your app. Because your self-improving React app will use the objective to assess its performance.

Let's say we want the users of our app to click on a certain button anywhere in our component hierarchy.

```
import react from 'react';
import { withObjective } from 'react-recommender';

export const Objective = withObjective(
  ({onAchieved, ...props}) => <button onClick={(evt)=> onAchieved("OptimizeClicks")}>
↪Click Me</button>
);
```

We import the `withObjective`-function from `react-recommender`. It adds the `onAchieved`-callback function to the properties of the wrapped component. We separate the `onAchieved`-property from the other properties by using the Spread Syntax (`{onAchieved, ...props}`).

Calling the `onAchieved`-function implies our app achieved an objective. In our example, a click on the button lets our app achieve the objective `OptimizeClicks`.

2.1.4 Define The Options

In order for our app to improve itself to achieve the objective, it must have the ability to change. We have to provide options to our React app. Options, the app can select one from when it renders. You can use the options to display different content, like your app's headline. You can apply different styles, like the color of a button. You can even use different options of your app's internal logic-if you like. You can use as an option whatever you can put into a React component. And that is pretty much anything you can think of.

```
import react from 'react';
import { Recommend, Option } from 'react-recommender';

export const SelectOption = (props) => (
  <Recommend
    mode="egreedy"
    epsilon={0.1}
    objectiveId="OptimizeClicks"
    options={[
      <Option id="helloWorld">
        <div>Hello World</div>
      </Option>,
      <Option id="helloYou">
        <div>Hello You</div>
      </Option>
    ]}>{
        ({loading, error, recommendation, renderOption}) => {
            return (loading && <div>Loading</div>) ||
                (recommendation ? renderOption(recommendation) : <div>Error</div>)
        }
    }
  </Recommend>
);
```

We provide two options. The first shows a `<div/>` saying `Hello World`. The second shows a `<div/>` saying `Hello You`. We wrap each of the options into an `<Option/>`-component. We provide the array of options as a property to a `<Recommend/>`-component.

When your React app renders this `<Recommend/>`-component, it checks which of the options promises to have the best chance of achieving the specified objective (here: `OptimizeClicks`).

We provide a function as the child of the `<Recommend/>`-component. This function takes a few parameters:

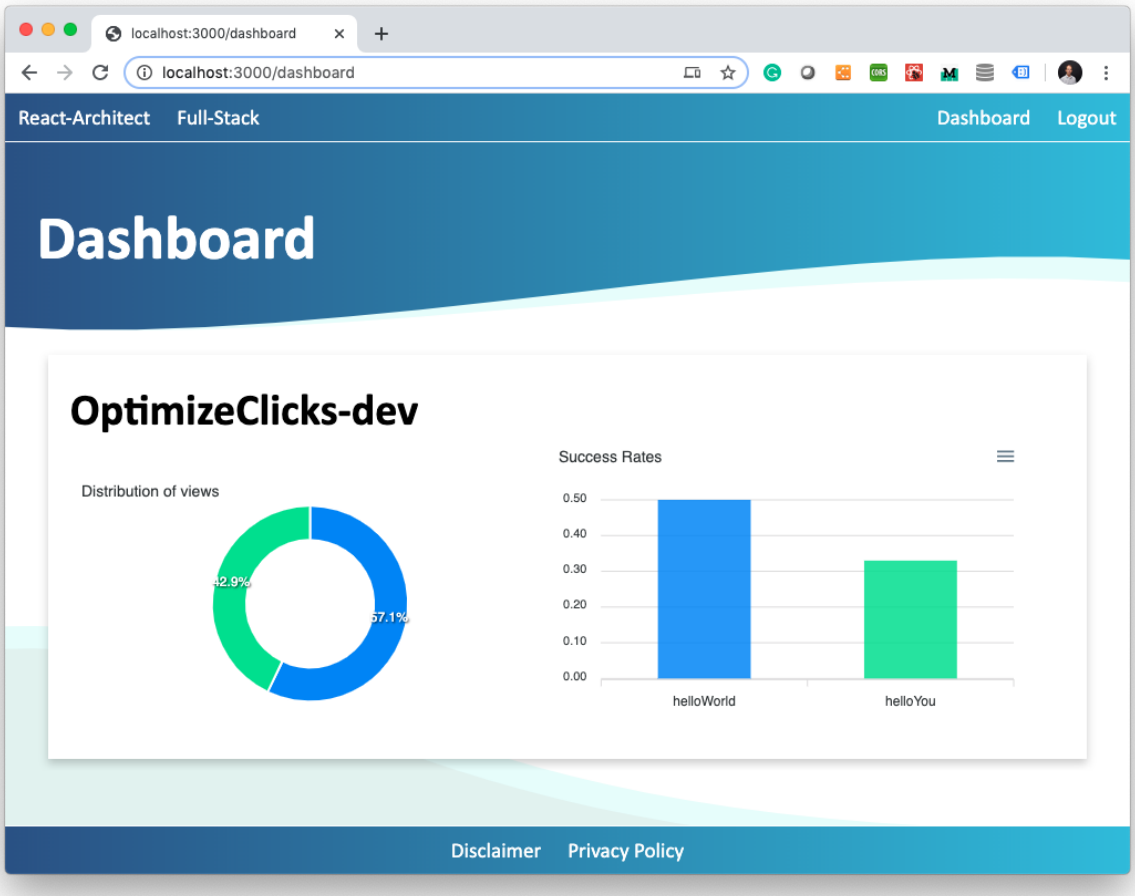
- `loading` is a `Boolean`-value. It indicates whether the app is currently loading the recommendation (`true`) or whether it has finished (`false`).
- `error` may contain an error message if something went wrong while loading the best option.
- If there was no error and loading finished, `recommendation` contains the `id` of the recommended option. You specify the `id` as the property of an `<Option/>`-component.
- You can use the `renderOption`-callback function to render the content of an option by specifying the

`<Option/>`'s `id`. Usually, this is the `id` you get as the `recommendation`. But you are free to overrule it.

This is all you need to build a self-improving React app.

2.1.5 Monitor The Improvements

`react-recommender` comes with a serverless backend. It counts how many times your app renders the options and achieves the objective. You can have a look at the current state of your recommender at https://www.react-architect.com.



2.2 Reference

2.2.1 Option

An `<Option/>`-component is a wrapper to be used in the `options`-property of reference/recommend-component.

See reference/option for more details.

2.2.2 Recommend

The `<Recommend/>`-component takes different options and recommends the best one.

See reference/recommend for more details.

2.2.3 Recommender

The `<Recommender/>`-component is the top-level integration.

See reference/recommender for more details.

2.2.4 withObjective

`withObjective` is a wrapper function you can use to define an objective.

See reference/withobjective for more details.