

Feature Function Overhaul

Michal Hrusecky, Tomas Caithaml, Chris Dyer

Accomplishments

- Feature functions are now handled in a engineer-friendly way:
- Stateless feature functions implement `StatelessFeatureFunction`
- Stateful feature functions implement `StatefulFeatureFunction`

Stateful Features

- Return arbitrary state after computing value
- Used to split DP states
- Passed as “previous” state to edges that derive (immediately) from that node
- Example: Language model, Distortion


```
class StatefulFeatureFunction: public FeatureFunction {  
  
public:  
  
    virtual void Evaluate(  
        const Hypothesis& cur_hypo,  
        const FFState* prev_state,  
        ScoreComponentCollection* scoreBreakdown,  
        FFState** cur_state) = 0;
```

State representation:

```
class FFState {  
public:  
    virtual ~FFState();  
    virtual int Compare(const FFState& other) const = 0;  
};
```

```
class StatefulFeatureFunction: public FeatureFunction {
```

```
public:
```

```
    virtual void Evaluate(  

```

```
        const Hypothesis& cur_hypo,  
        const FFState* prev_state,  
        ScoreComponentCollection* scoreBreakdown,  
        FFState** cur_state) = 0;
```

State representation:

```
class FFState {
```

```
public:
```

```
    virtual ~FFState();
```

```
    virtual int Compare(const FFState& other) const = 0;
```

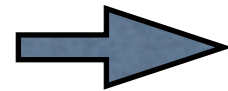
```
};
```

```
class StatefulFeatureFunction: public FeatureFunction {  
  
public:  
  
    virtual void Evaluate(  
        const Hypothesis& cur_hypo,  
        const FFState* prev_state,  
        ScoreComponentCollection* scoreBreakdown,  
        FFState** cur_state) = 0;
```

State representation:

```
class FFState {  
public:  
    virtual ~FFState();  
    virtual int Compare(const FFState& other) const = 0;  
};
```

```
class StatefulFeatureFunction: public FeatureFunction {  
  
public:  
  
    virtual void Evaluate(  
        const Hypothesis& cur_hypo,  
        const FFState* prev_state,  
        ScoreComponentCollection* scoreBreakdown,  
        FFState** cur_state) = 0;
```



State representation:

```
class FFState {  
public:  
    virtual ~FFState();  
    virtual int Compare(const FFState& other) const = 0;  
};
```

```
class StatefulFeatureFunction: public FeatureFunction {  
  
public:  
  
    virtual void Evaluate(  
        const Hypothesis& cur_hypo,  
        const FFState* prev_state,  
        ScoreComponentCollection* scoreBreakdown,  
        FFState** cur_state) = 0;
```



State representation:

```
class FFState {  
public:  
    virtual ~FFState();  
    virtual int Compare(const FFState& other) const = 0;  
};
```



Stateless Features

- Compute the same value *independent* of context
- Generation scores
- Translation scores
- Word penalty

```
class StatelessFeatureFunction: public FeatureFunction {
```

```
public:
```

```
    virtual void Evaluate(
```

```
        const TargetPhrase& cur_hypo,  
    ScoreComponentCollection* out) = 0;
```

```
class StatelessFeatureFunction: public FeatureFunction {
```

```
public:
```

```
    virtual void Evaluate(
```

```
        const TargetPhrase& cur_hypo,
```

```
        ScoreComponentCollection* out) = 0;
```

Continuing work

- New feature functions!
- Removing dependencies on *specific* feature types from n-best extraction, command line interface, MERT support scripts

Děkujeme vám!