Sparse Features in Moses

Colin Cherry
Barry Haddow

7th September, 2012

Timings

- Using core features
  - Trunk Moses: 47 minutes
  - Sparse Moses: 54 minutes
- Using extended features (and a smaller model)
  - Sparse Moses: 372/377 minutes
  - Sparse Moses (pre-calculation): 63/65 minutes
  - Sparse Moses (eliminate gratuitous waste): 58 minutes

Timings

- Using core features
  - Trunk Moses: 47 minutes
  - Sparse Moses: 54 minutes
- Using extended features (and a smaller model)
  - Sparse Moses: 372/377 minutes
  - Sparse Moses (pre-calculation): 63/65 minutes
  - Sparse Moses (eliminate gratuitous waste): 58 minutes

Timings

- Using core features
  - Trunk Moses: 47 minutes
  - Sparse Moses: 54 minutes
- Using extended features (and a smaller model)
  - Sparse Moses: 372/377 minutes
  - Sparse Moses (pre-calculation): 63/65 minutes
  - Sparse Moses (eliminate gratuitous waste): 58 minutes

Optimisations I

- Pre-calculate features which only depend on phrase pair / rule
  - "Stateless features" can use rule, source sentence and coverage
  - Those in phrase table already pre-calculated
  - Added hooks for pre-calculating others
- Reduce (or eliminate!) string concatenation in names
  - Use "builder" object for feature names
  - Standard version will encode strings
  - Fast version can use hashing

Optimisations I

- Pre-calculate features which only depend on phrase pair / rule
  - "Stateless features" can use rule, source sentence and coverage
  - Those in phrase table already pre-calculated
  - Added hooks for pre-calculating others
- Reduce (or eliminate!) string concatenation in names
  - Use "builder" object for feature names
  - Standard version will encode strings
  - Fast version can use hashing

Optimisations I

- Pre-calculate features which only depend on phrase pair / rule
  - "Stateless features" can use rule, source sentence and coverage
  - Those in phrase table already pre-calculated
  - Added hooks for pre-calculating others
- Reduce (or eliminate!) string concatenation in names
  - Use "builder" object for feature names
  - Standard version will encode strings
  - Fast version can use hashing

Optimisations II

- Don't store full feature vectors unless needed
  - Add a method to feature function to calculate score delta
  - For n-best lists, backtrack to get vectors
- Use a faster feature vector
  - Take from cdec, or from kbmira
  - Moses should have just one – it has 3 now

Optimisations II

- Don't store full feature vectors unless needed
  - Add a method to feature function to calculate score delta
  - For n-best lists, backtrack to get vectors
- Use a faster feature vector
  - Take from cdec, or from kbmira
  - Moses should have just one – it has 3 now

Optimisations II

- Don't store full feature vectors unless needed
  - Add a method to feature function to calculate score delta
  - For n-best lists, backtrack to get vectors
- Use a faster feature vector
  - Take from cdec, or from kbmira
  - Moses should have just one – it has 3 now