Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Phrase-based Baselines

[Ziemski et al., 2016] provided BLEU scores for Moses configurations trained on the 6-way subcorpus.

- Word alignment with MIGZA++, 5x Model1, 5x HMM;
- 5-gram language model from the target parallel data;
- Default lexical reordering model;
- 5-gram operation sequence model;
- 9-gram word-class language model with word-classes produced by word2vec [Mikolov et al., 2013];
- Significance pruned [Johnson et al., 2007] phrase-table;
- For decoding, cube-pruning algorithm with stack size and cube-pruning pop limits of 1,000.

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

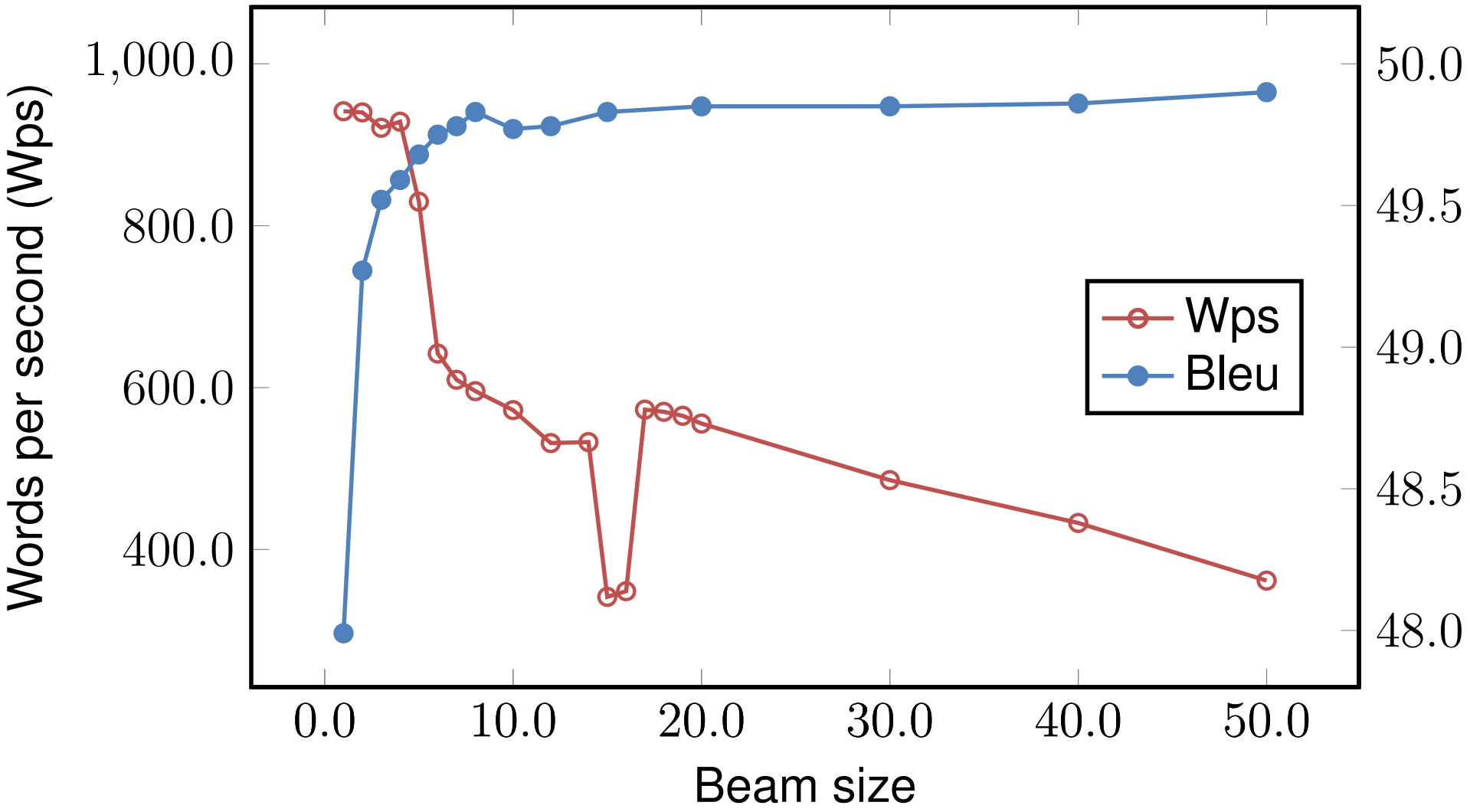
[2] `https://github.com/emjotde/amunmt`

Efficient decoding with AmuNMT

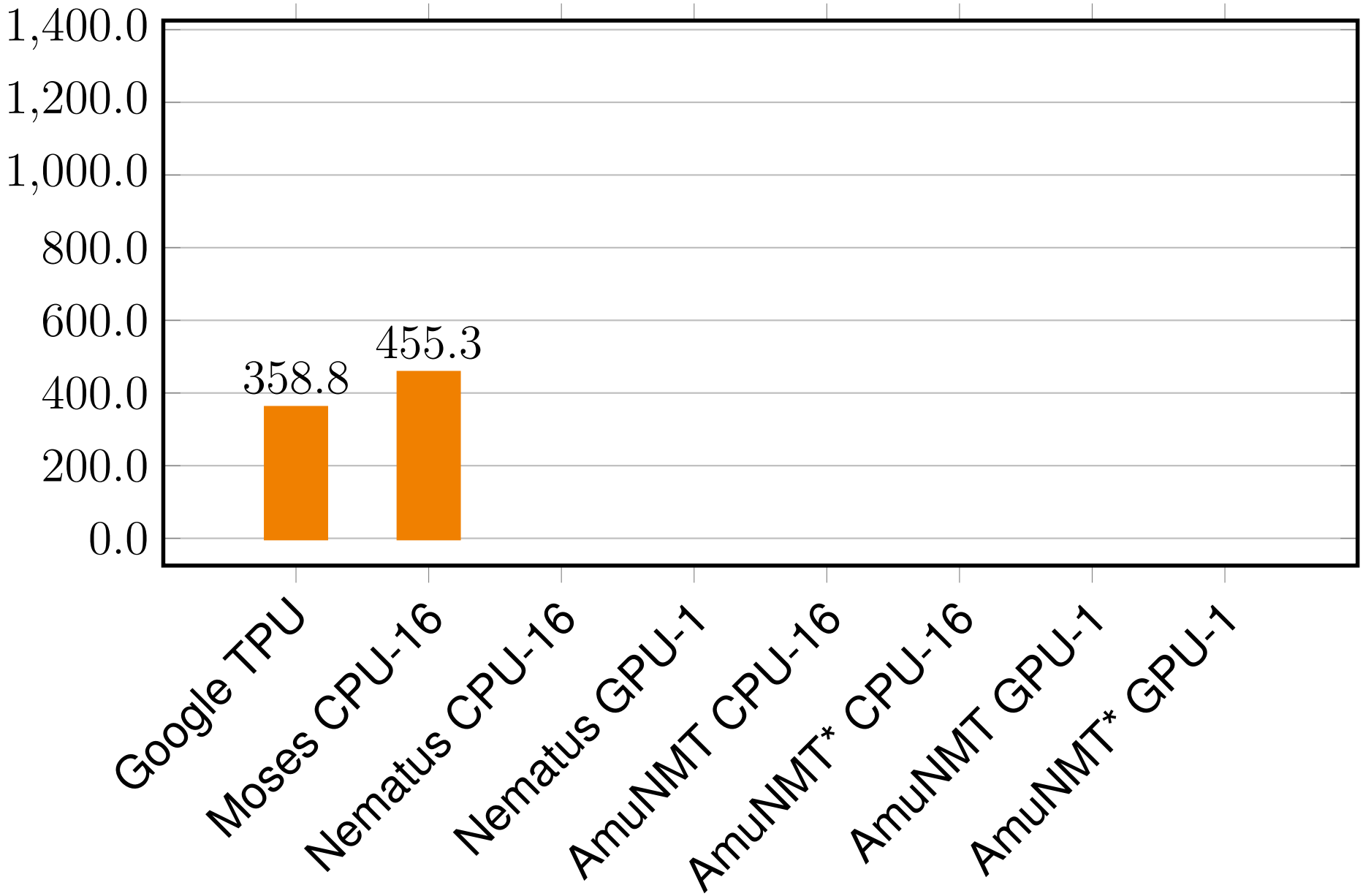
- AmuNMT[2] is a ground-up neural MT toolkit implementation, developed in C++;
- Grew out of a Moses feature function;
- Currently only inference (training framework in preparation);
- Compatible with Nematus [Sennrich et al., 2016];
- Multi-GPU, multi-CPU and mixed GPU/CPU mode with sentence-wise threads;
- Low-latency CPU-based decoding with intra-sentence multi-threading;
- Ensembling of multiple models;
- Vocabulary selection [Jean et al., 2015, Mi et al., 2016];
- Integrated segmentation into subwords [Sennrich et al., 2015].

[2] `https://github.com/emjotde/amunmt`

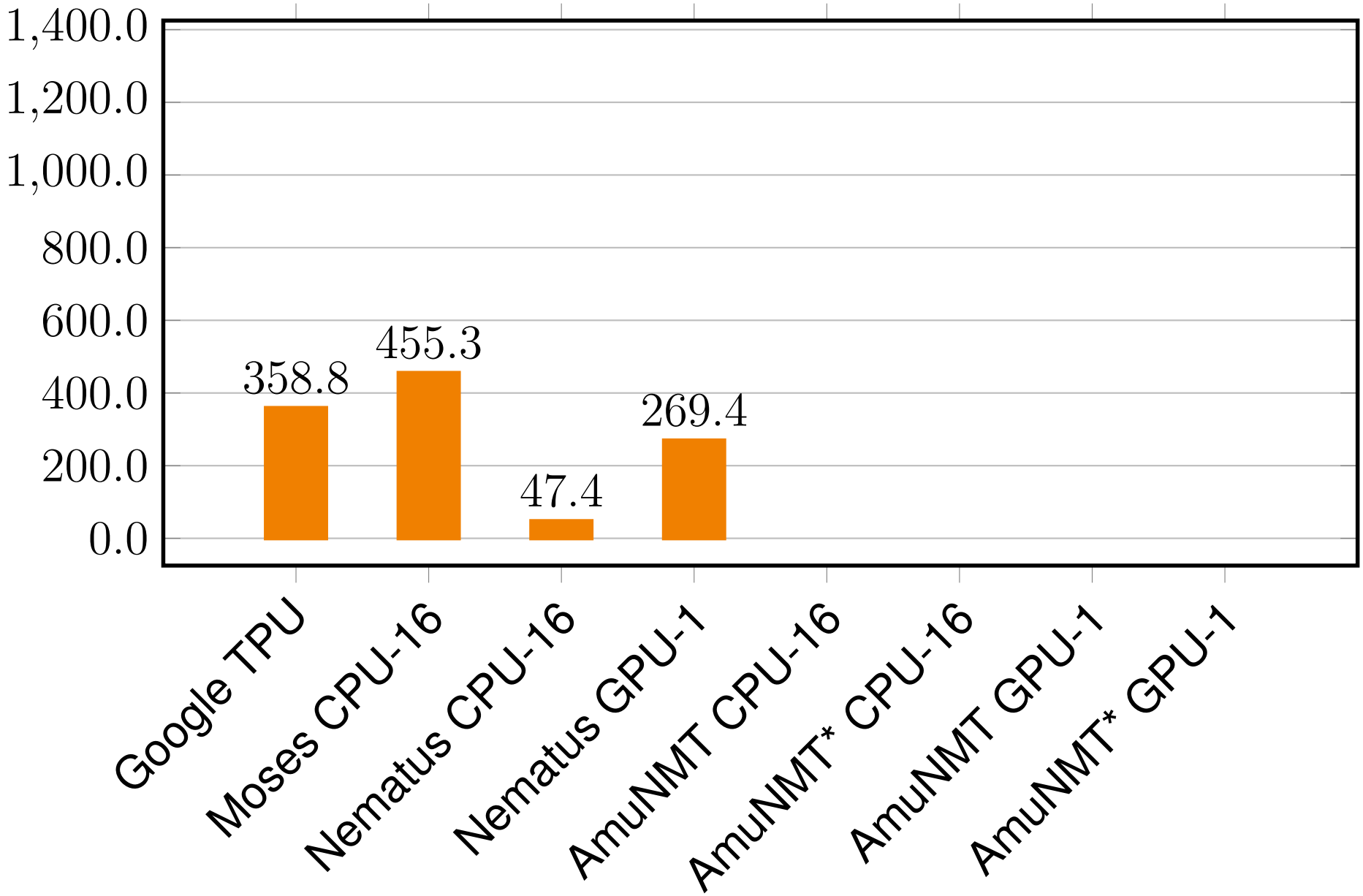
Speed and quality vs. beam size



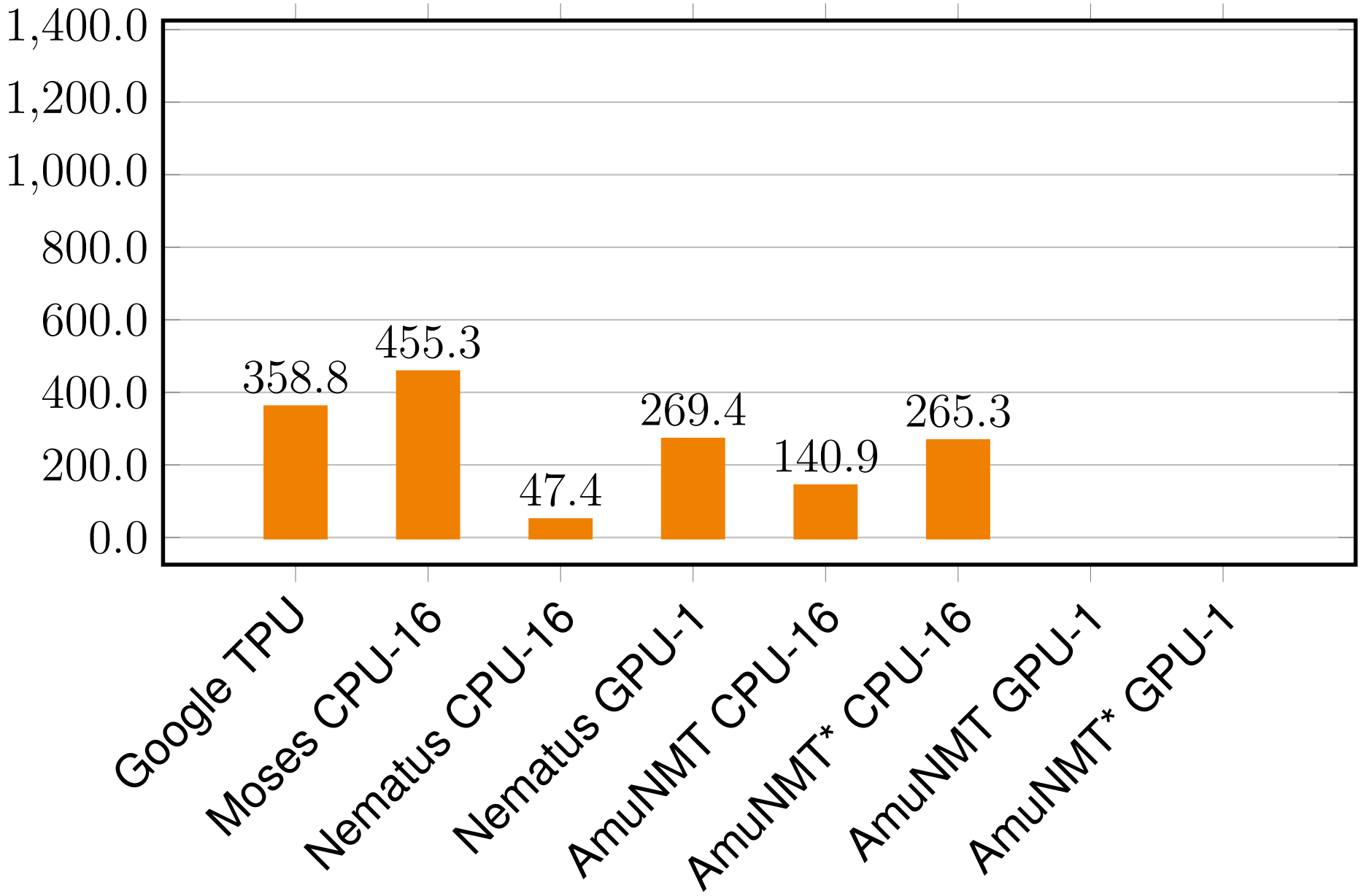
Translation speed in words per second



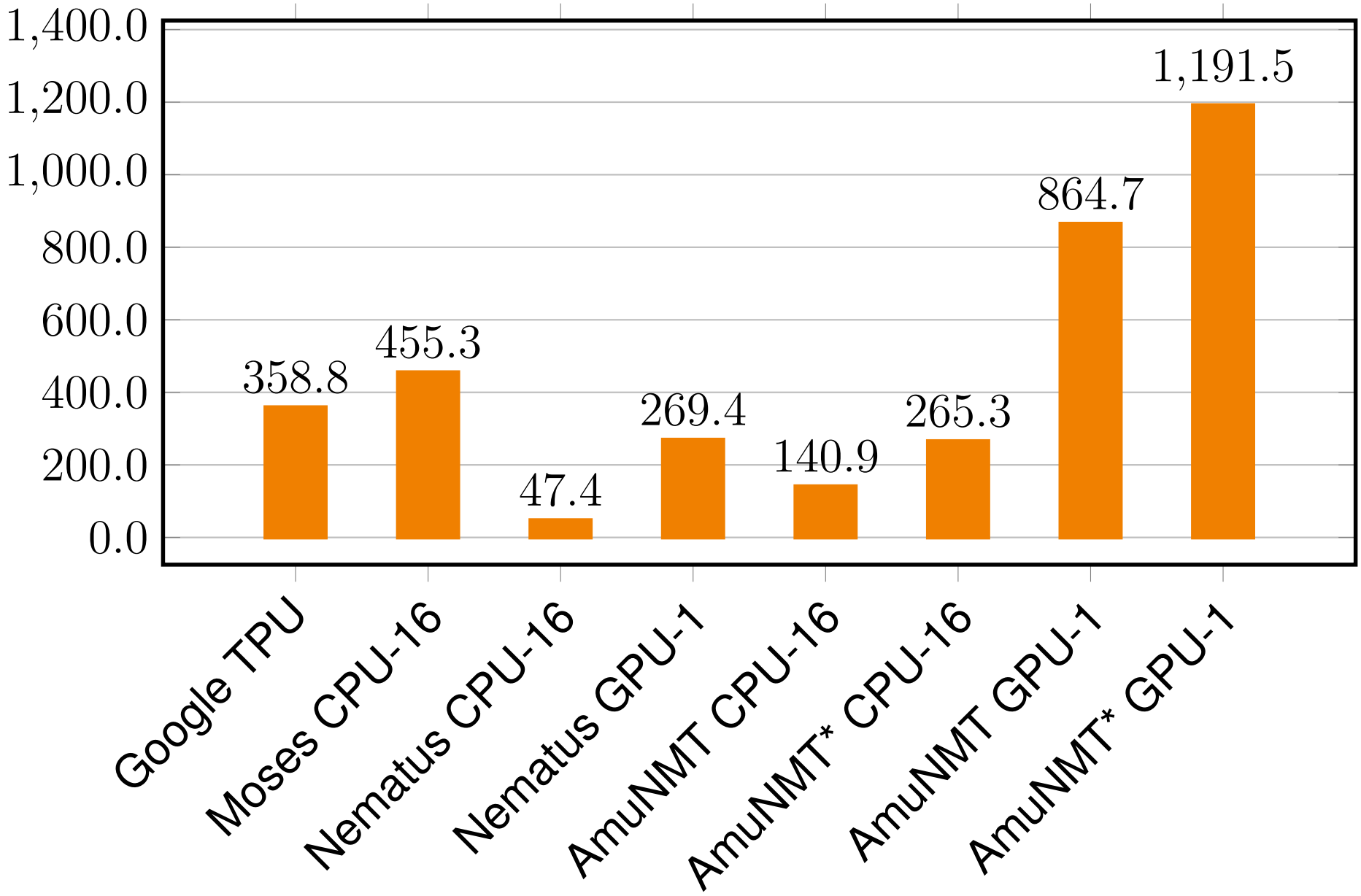
Translation speed in words per second



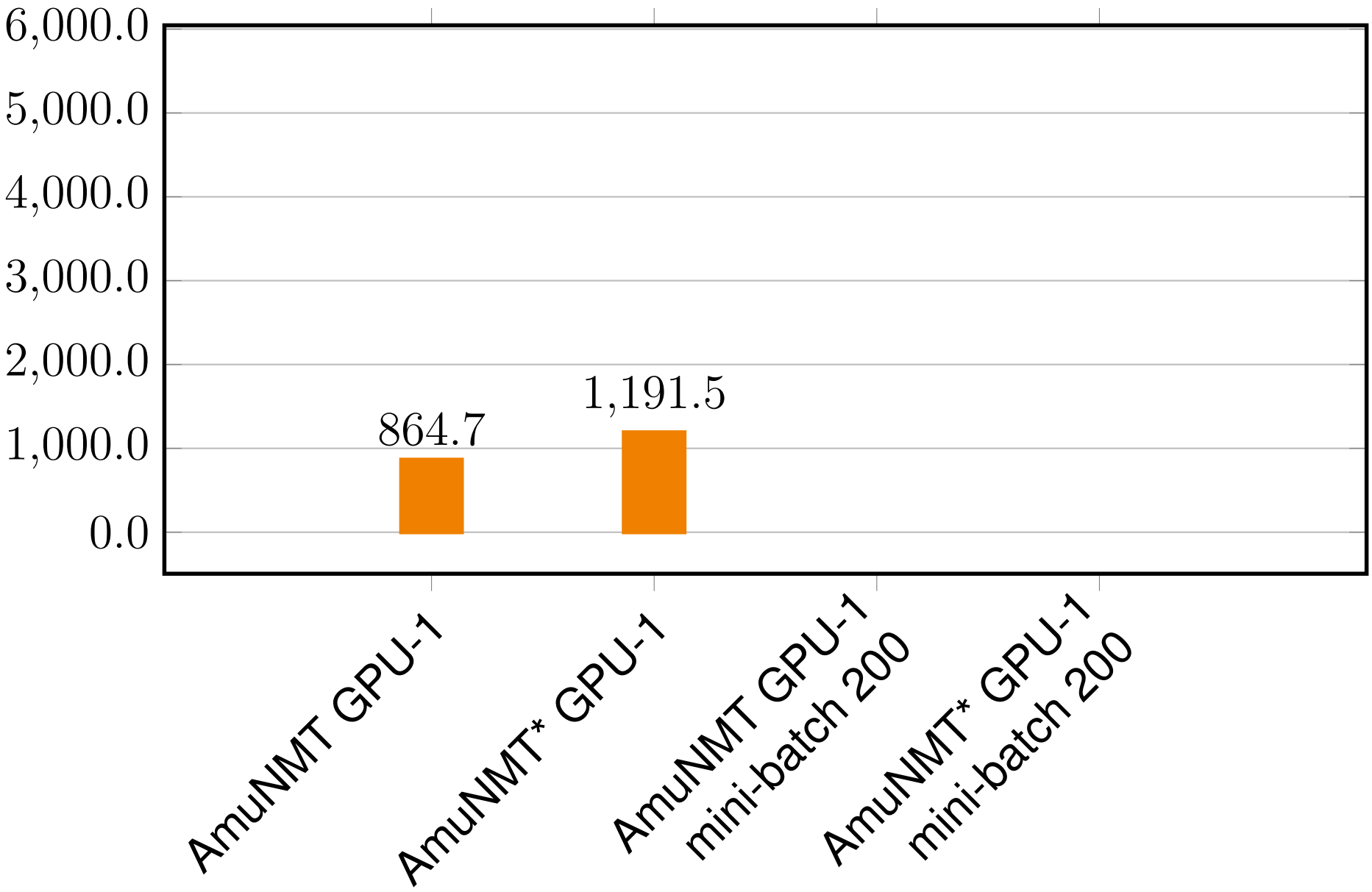
Translation speed in words per second



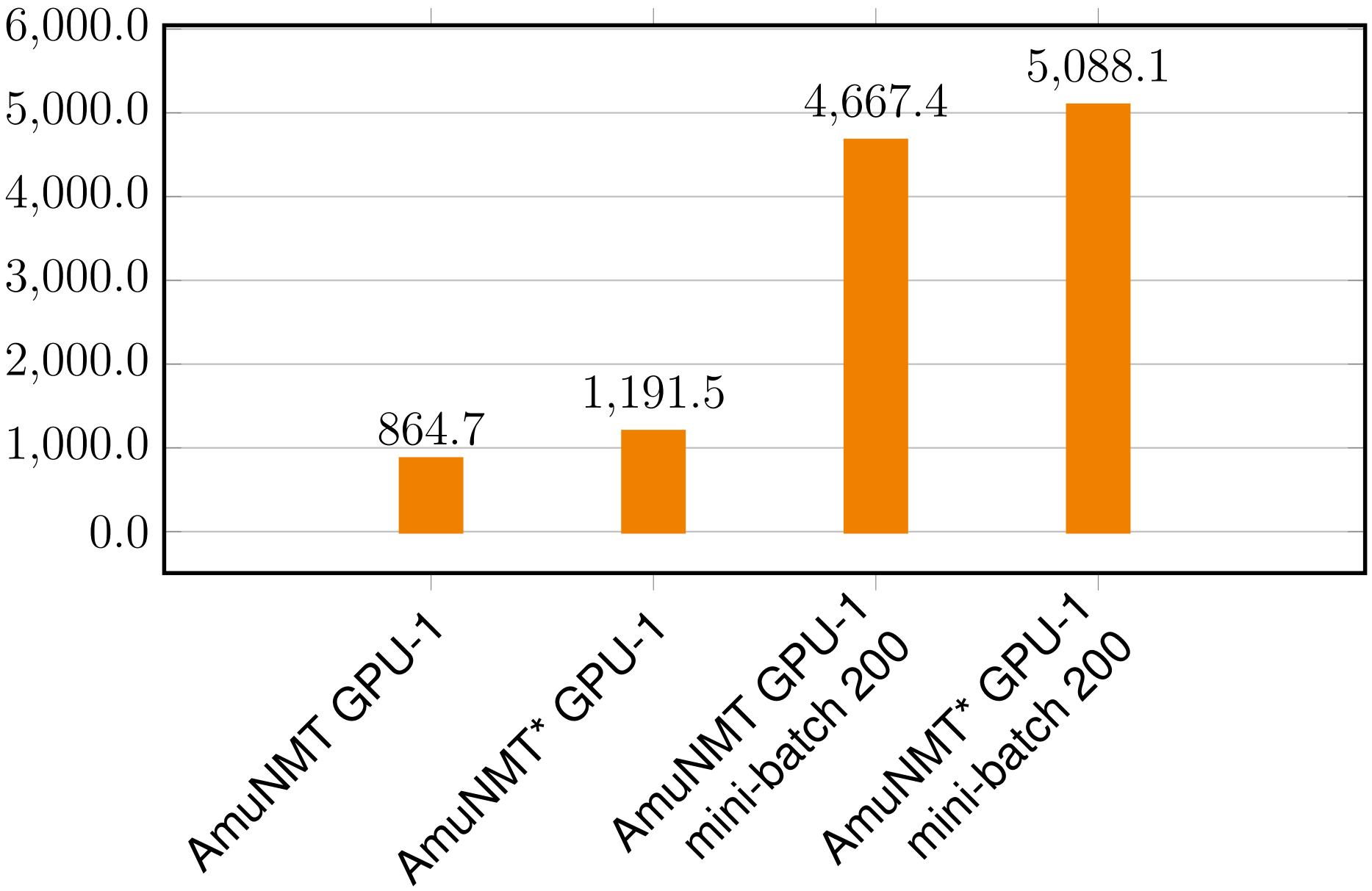
Translation speed in words per second



New: Batched decoding on the GPU

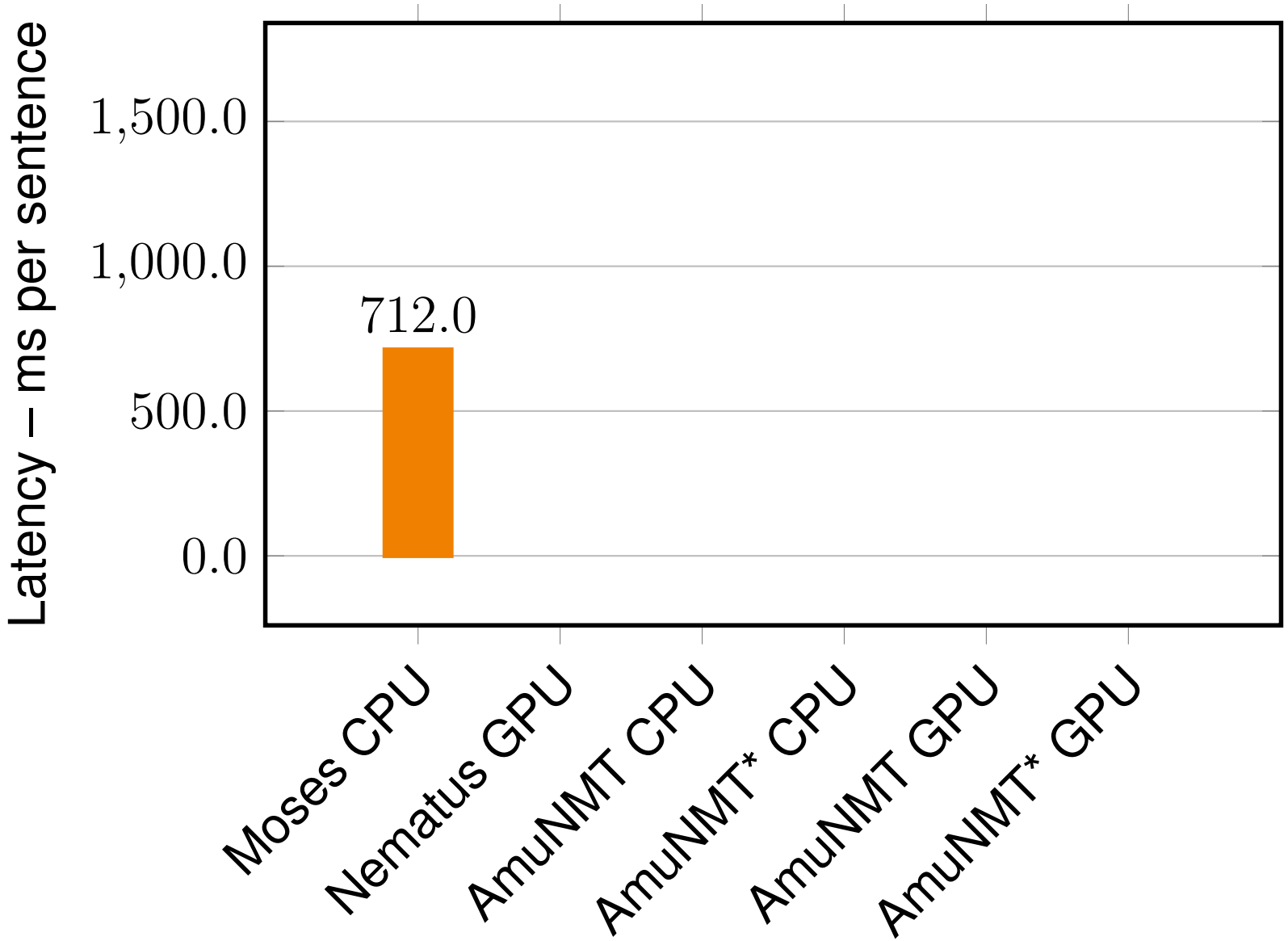


New: Batched decoding on the GPU



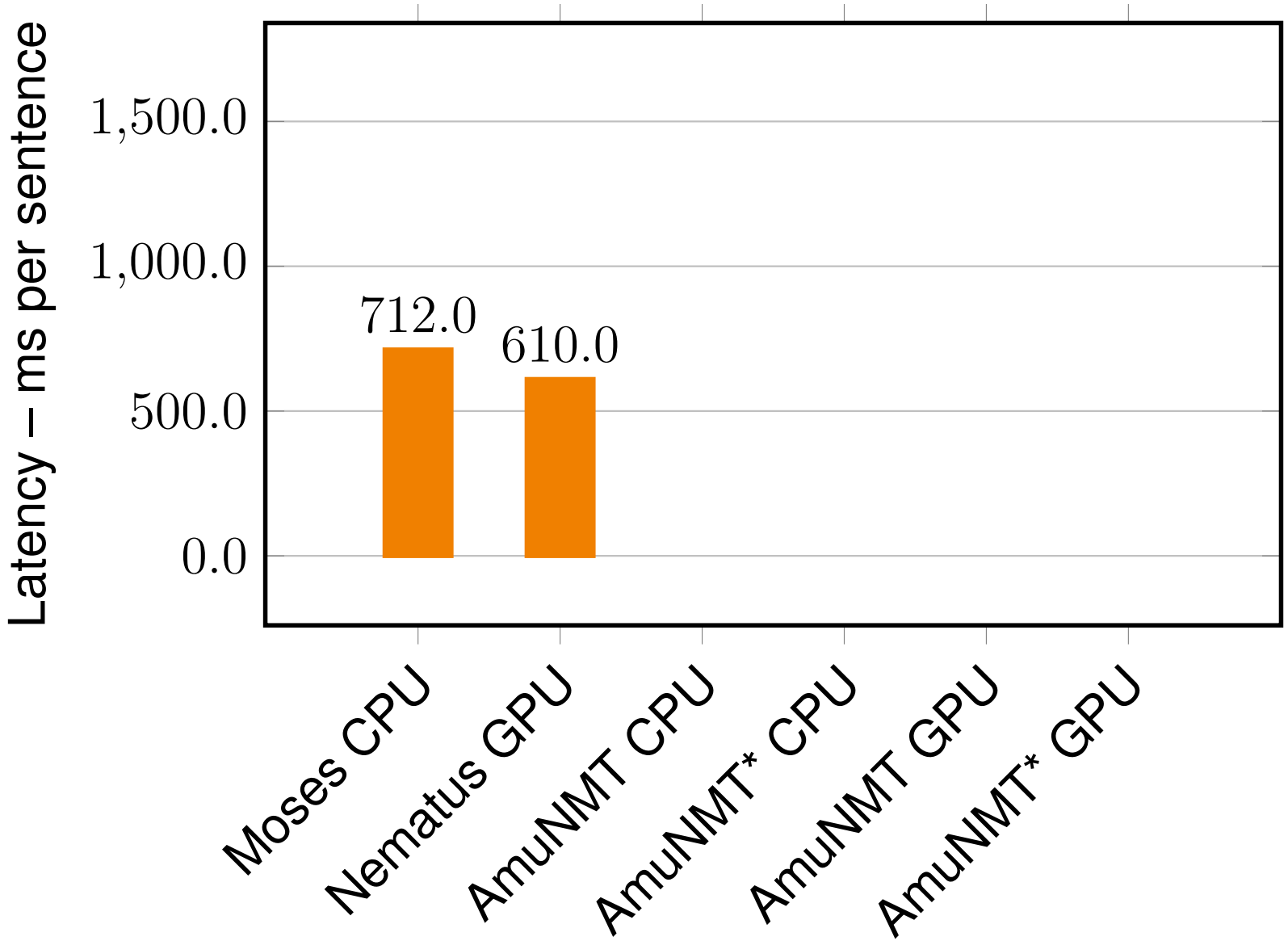
Latency in ms per sentence

Lower is better



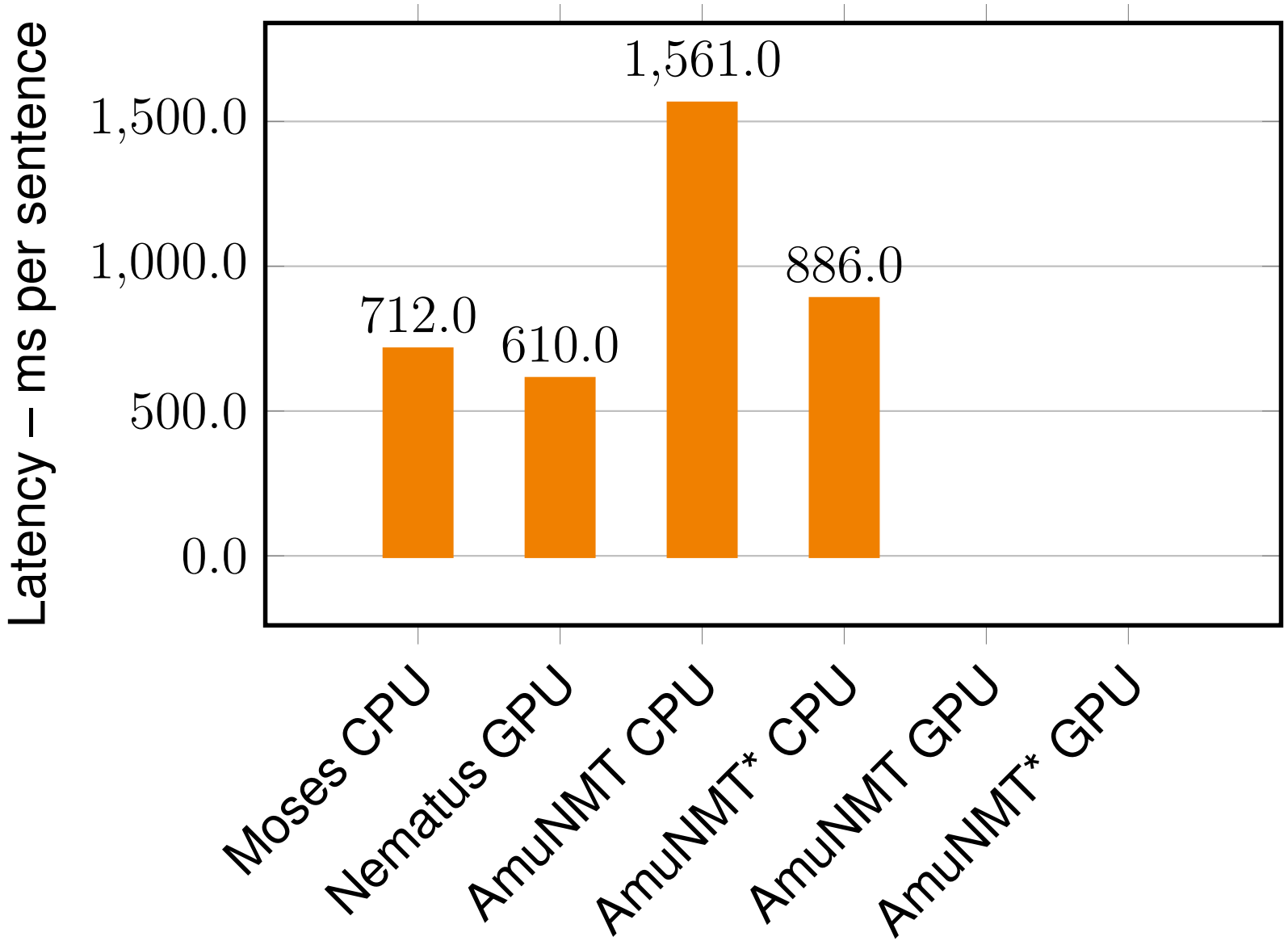
Latency in ms per sentence

Lower is better



Latency in ms per sentence

Lower is better



Latency in ms per sentence

Lower is better