Berkeley Parlab

Parallelizing a Statistical Machine Translator

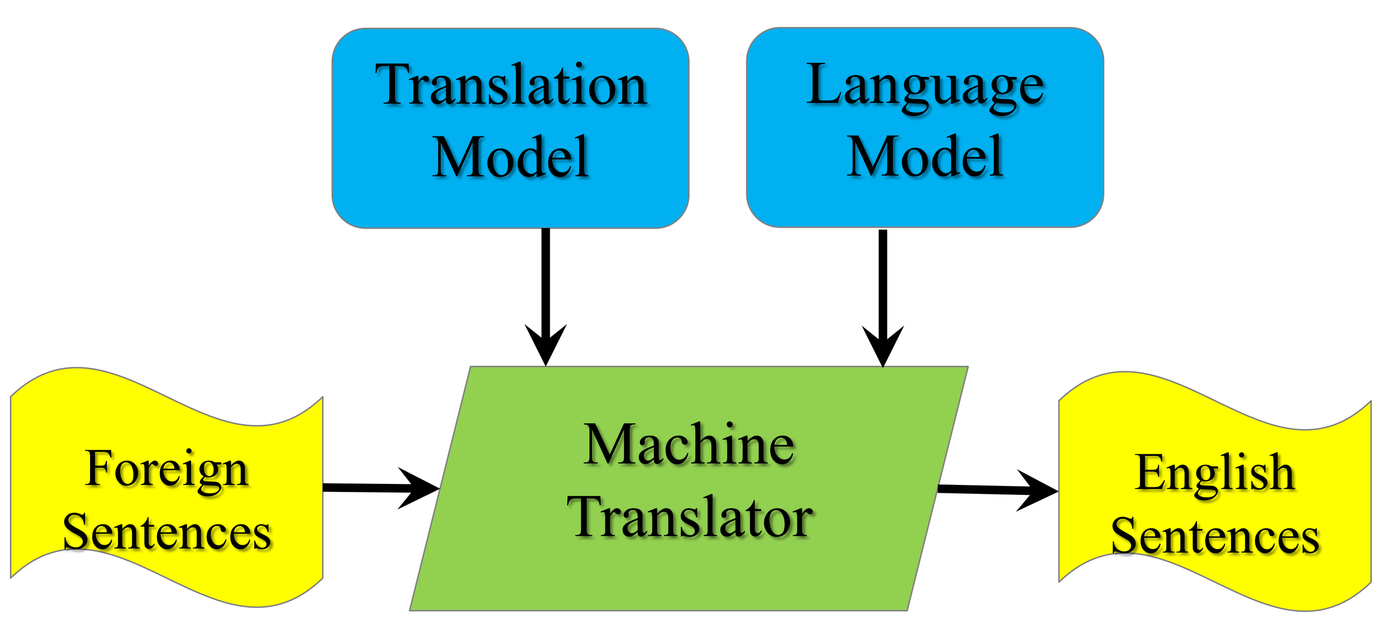
Chao-Yue Lai, Katya Gonina, Kurt Keutzer

Machine Translation

- High-quality translation is important.

- But high-quality translation: minutes for a 50-word sentence.
  - Huge language models
  - Complicated algorithm
- Need for speedups
  - Tons of texts on the Internet
  - Enable read-time translators

Overview of a Statistical Machine Translator



- Models: statistical information gathered from a large set of written documents
- Translation Models

C'est la vie. → This is life. : 50%
→ This is the life. : 30%
→ This is awful. : 10%

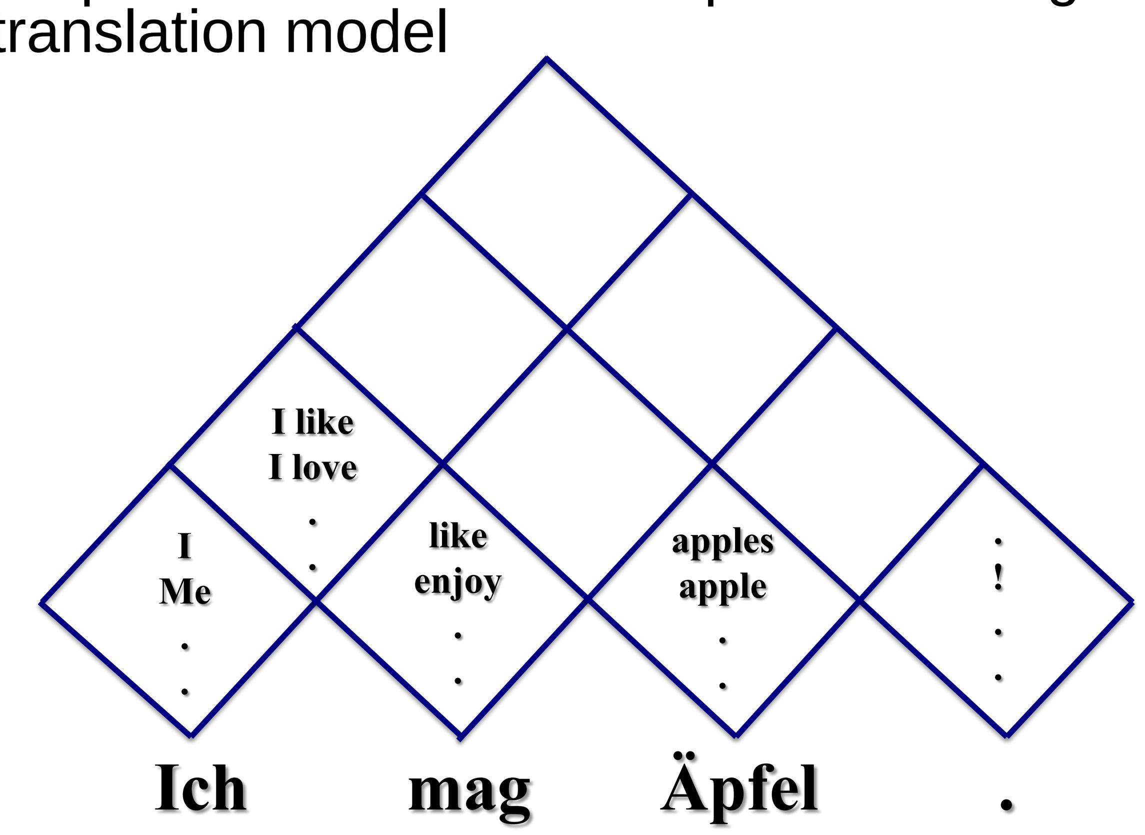
- N-gram Language Models

By the ? → way : 65%
→ run : 1%

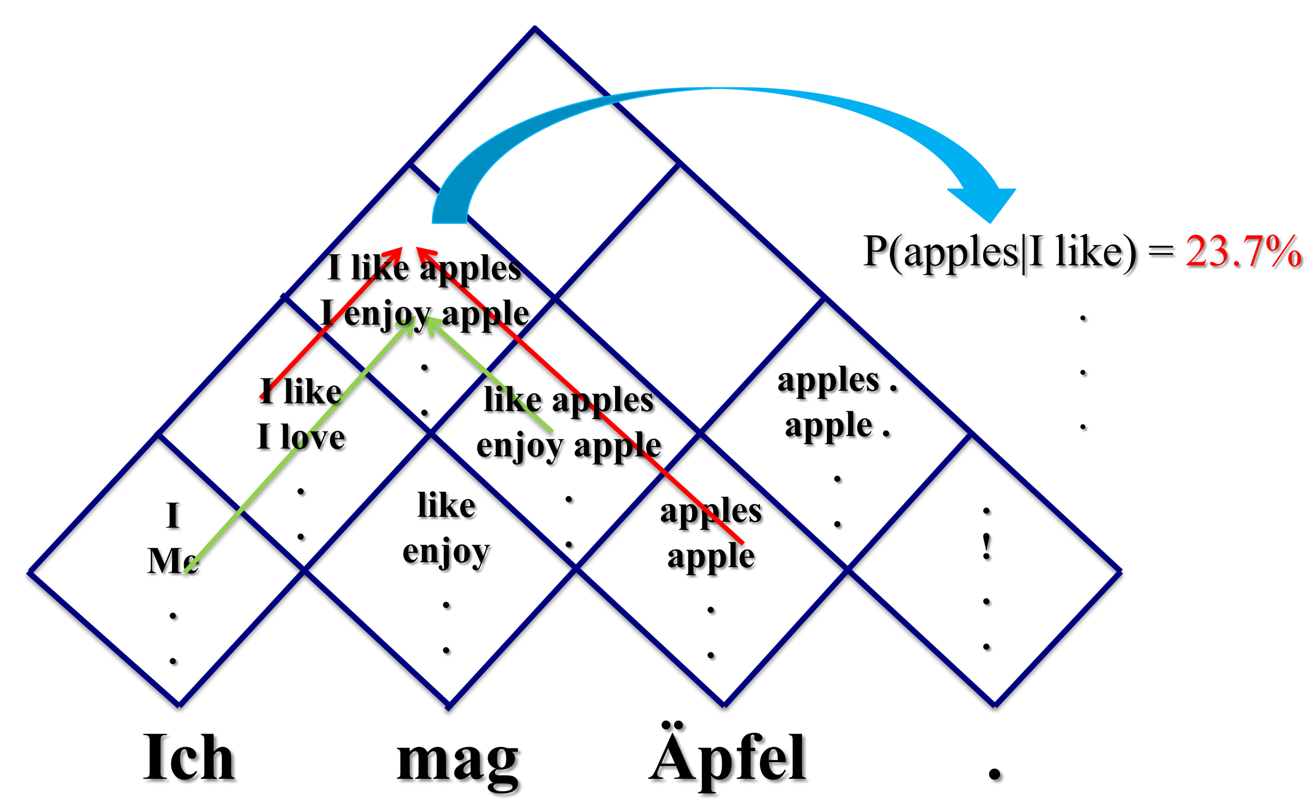
  - P(N-th word | preceding (N-1) words)
  - N can be up to 10 for quality

How it Works

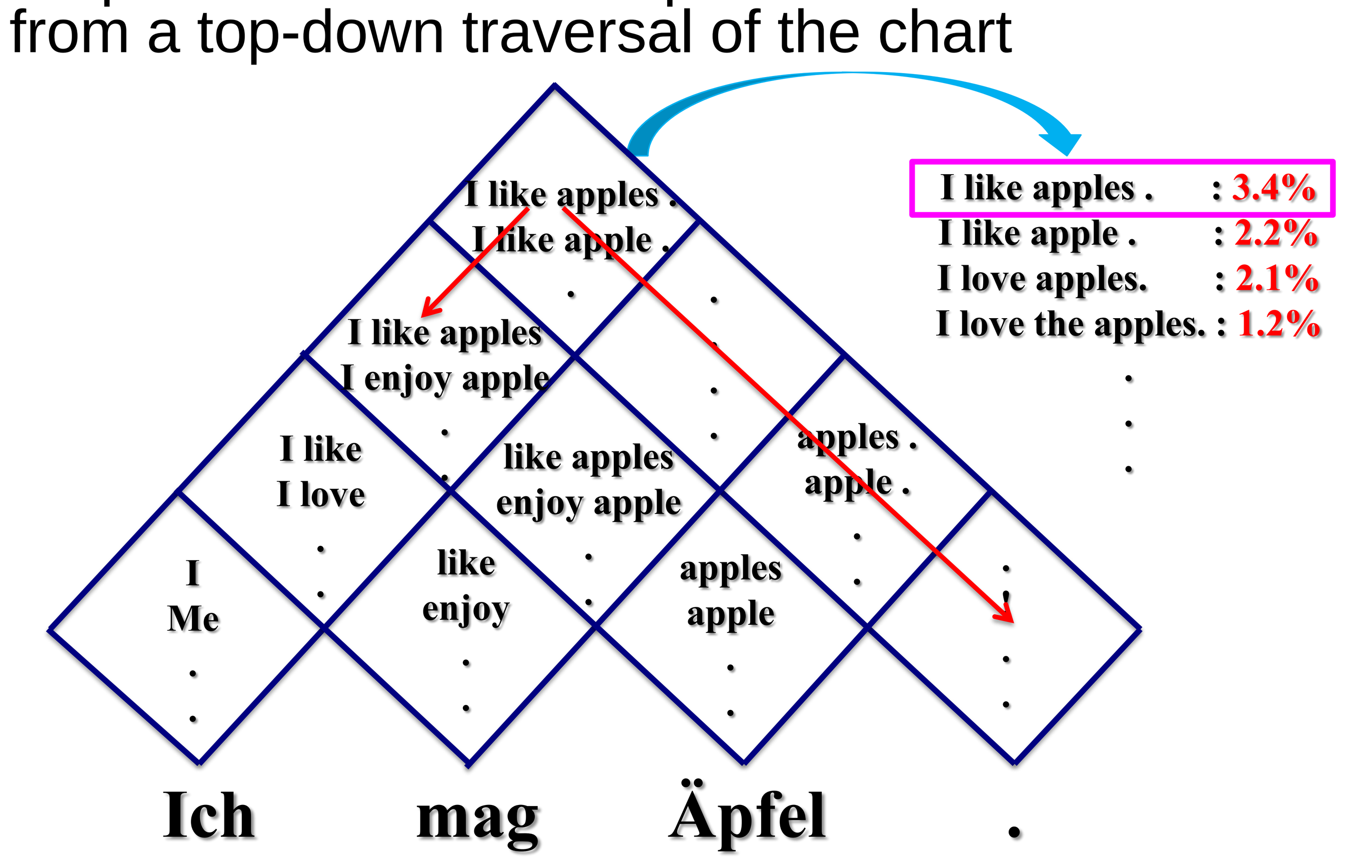
- Step 1: find translations of phrases using the translation model



- Step 2: combine translations and fill the chart bottom-up-wise using the language model



- Step 3: extract the most probable translation from a top-down traversal of the chart

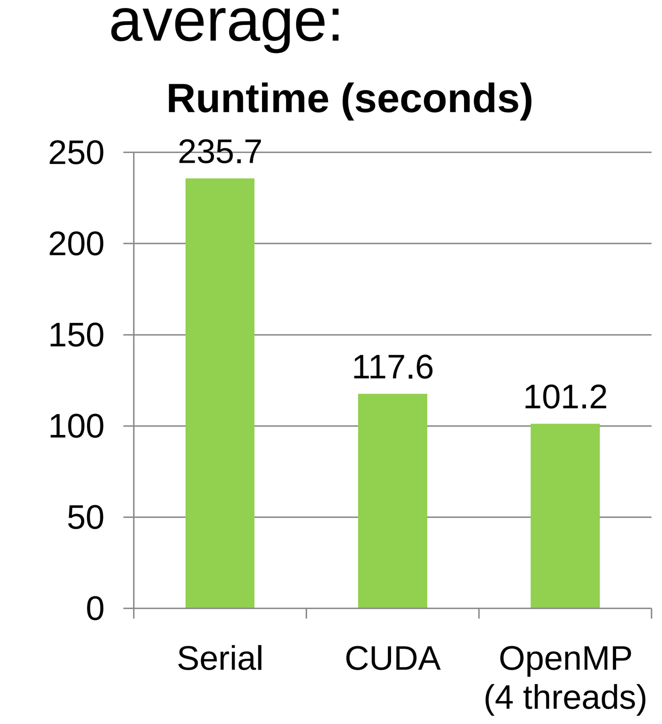


Parallelization Challenges

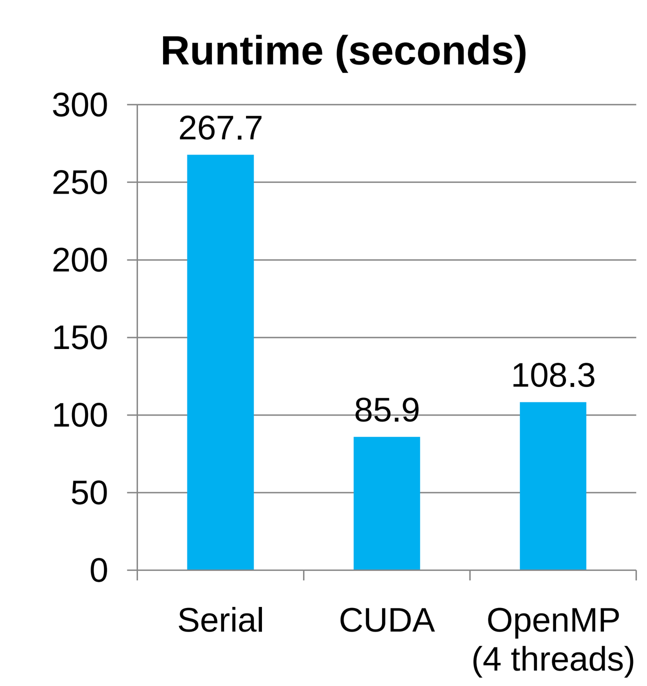
- Performance critically depends on efficient access of the language model
  - Evaluation of each phrase probabilities corresponds to irregular traversal of the language model graph
- Algorithmic similarity with the Speech App
  - Irregular access of language model
  - Iterative probabilistic inference & Pruning
  - However, even less "numeric"
- Accuracy highly depends on the size of the language model
  - Bigram → N-gram
  - Vocabulary size

Preliminary Results

- 1000 sentences (Spanish → English) with 28 words in average:



- GPU performs better in longer sentences (#words > 40)



Future Work

- Extract language model probability computation techniques from the speech app
- Bigram → N-gram, transform the language model for efficient access on the GPU
- Experiment with larger, more sophisticated machine translators (Berkeley Translator)