FULL-W2V: Fully Exploiting Data Reuse for Word2Vec on GPU-Accelerated Systems

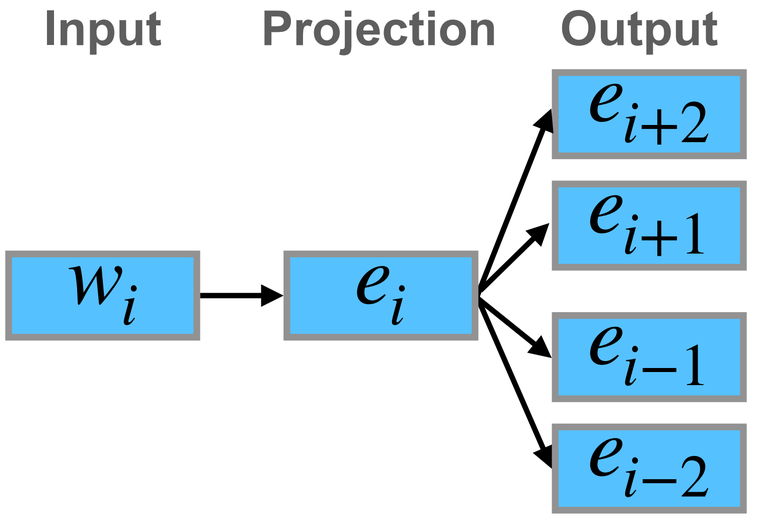
Thomas Randall, Tyler Allen, Rong Ge
{tlranda, tnallen, rge}@clemson.edu

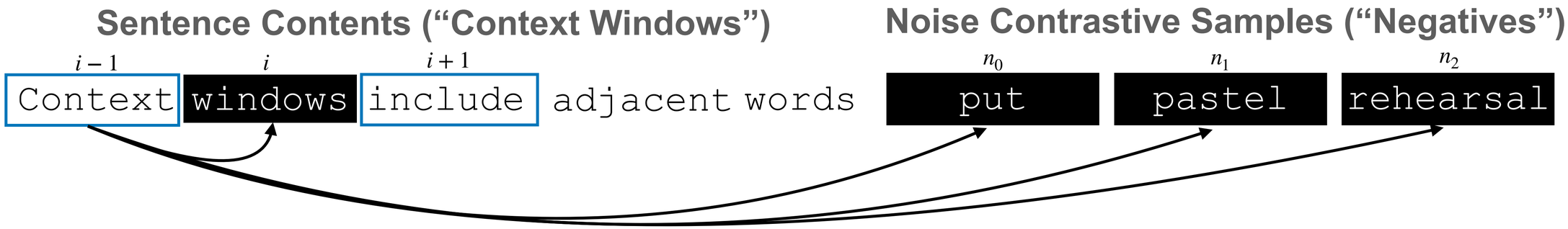
ICS '21, Virtual Event; June 17, 2021

Funded in part by NSF Grants:
CCF-1551511 and CNS-1551262

GPU W2V Not Faster than CPU

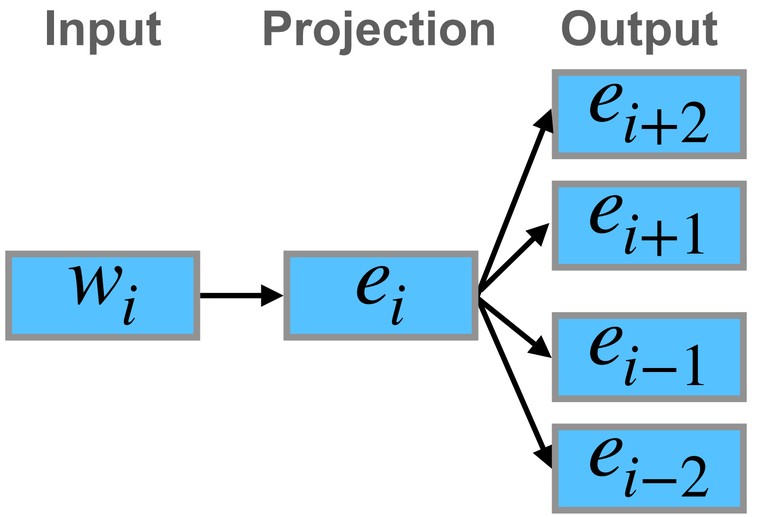
- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings e
- Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy

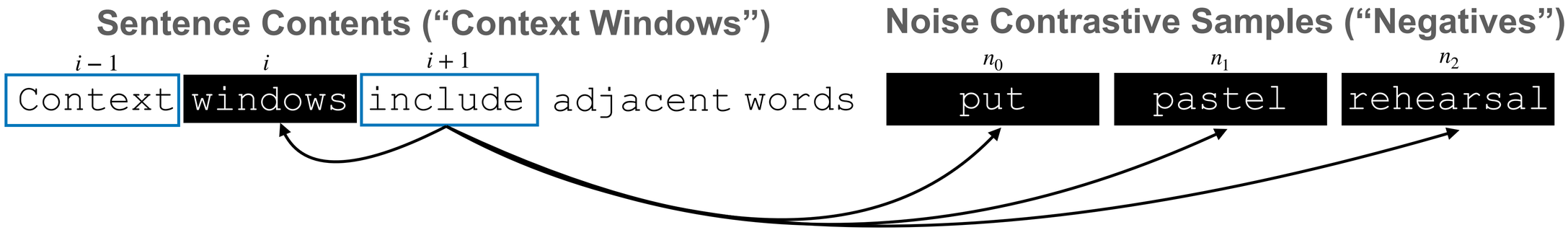




GPU W2V Not Faster than CPU

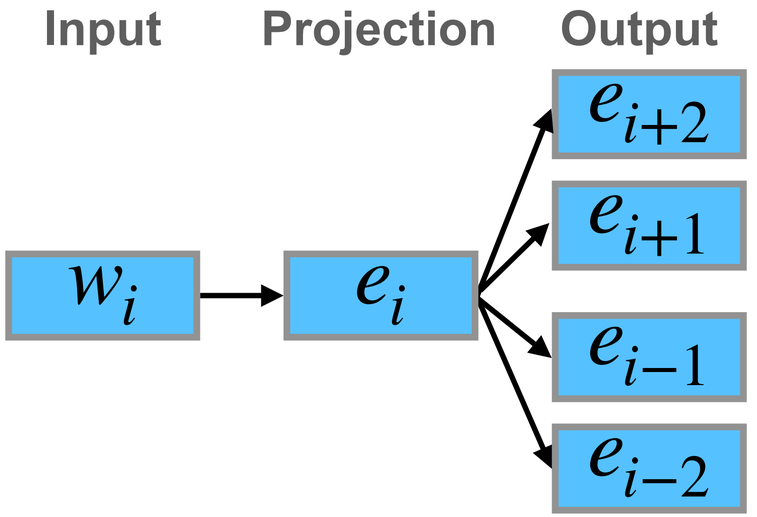
- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings e
- Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy

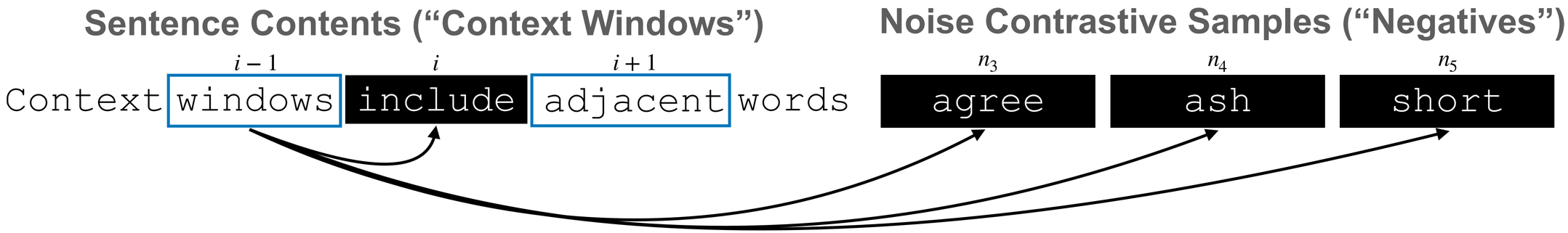




GPU W2V Not Faster than CPU

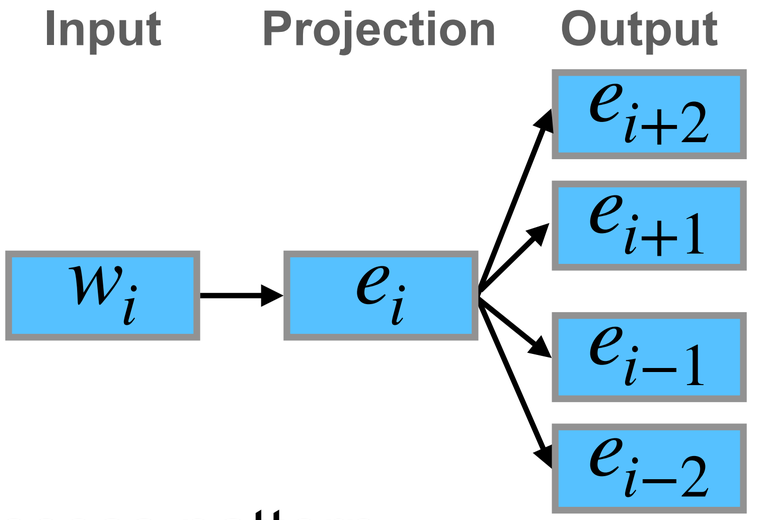
- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings e
- Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy

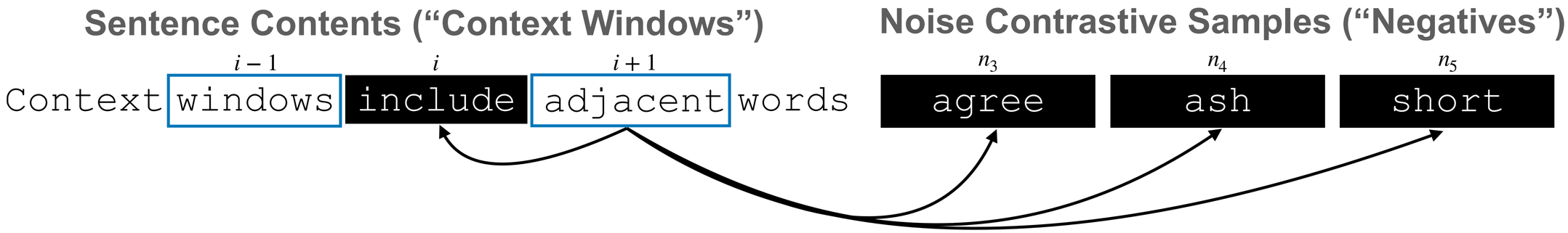




GPU W2V Not Faster than CPU

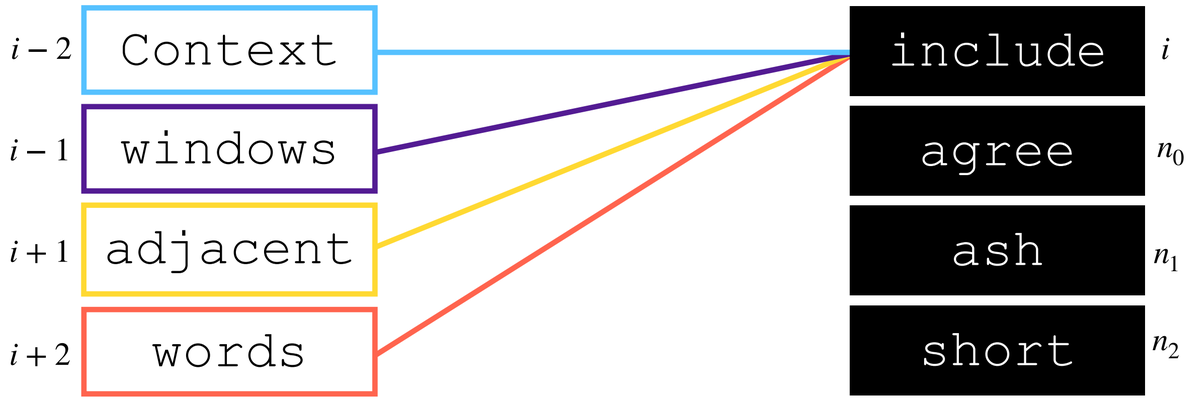
- 3-layer ANN
  - Words $w \rightarrow d$-dimensional embeddings e
- Prior ports based on data-intensive implementation
  - Suboptimal usage of GPU memory hierarchy
- FULL-W2V reduces access and improves locality
  - Leverage memory hierarchy based on algorithm's access pattern





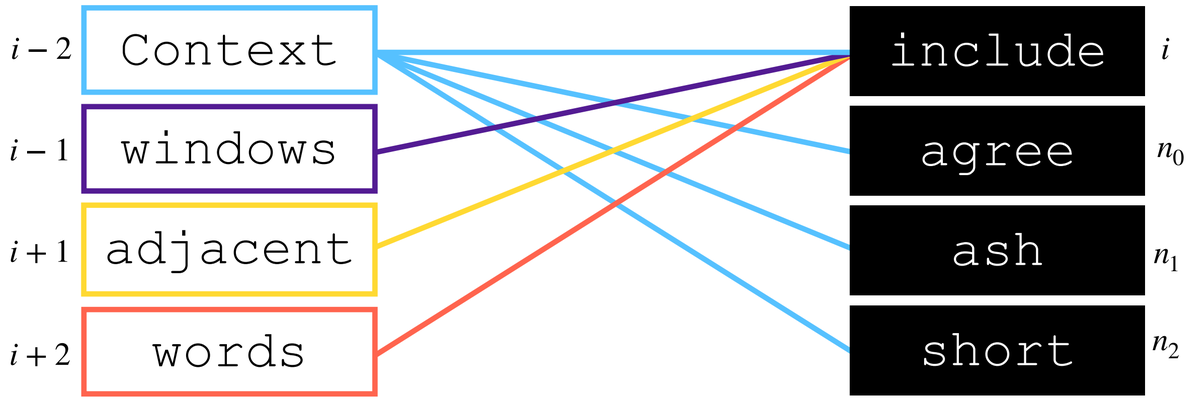
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**



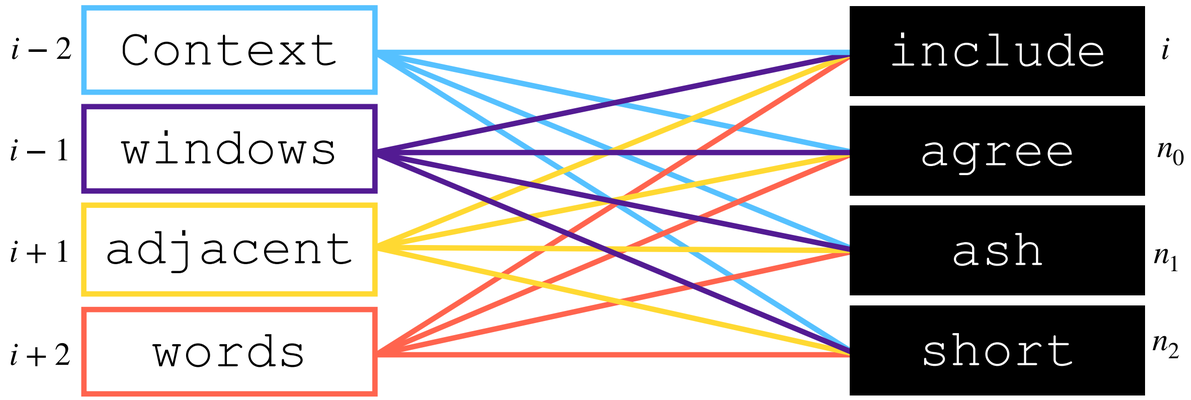
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**
- Opportunity: Operations have **independent order**



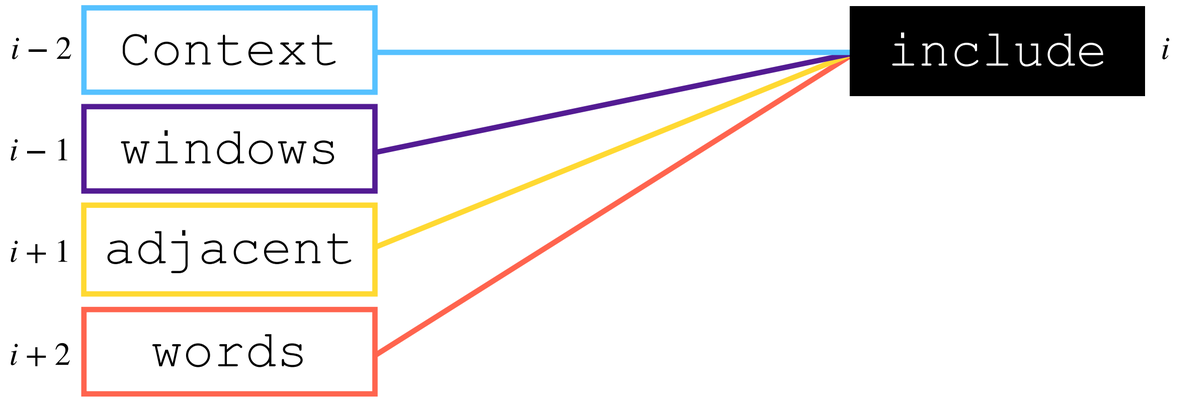
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**
- Opportunity: Operations have **independent order**



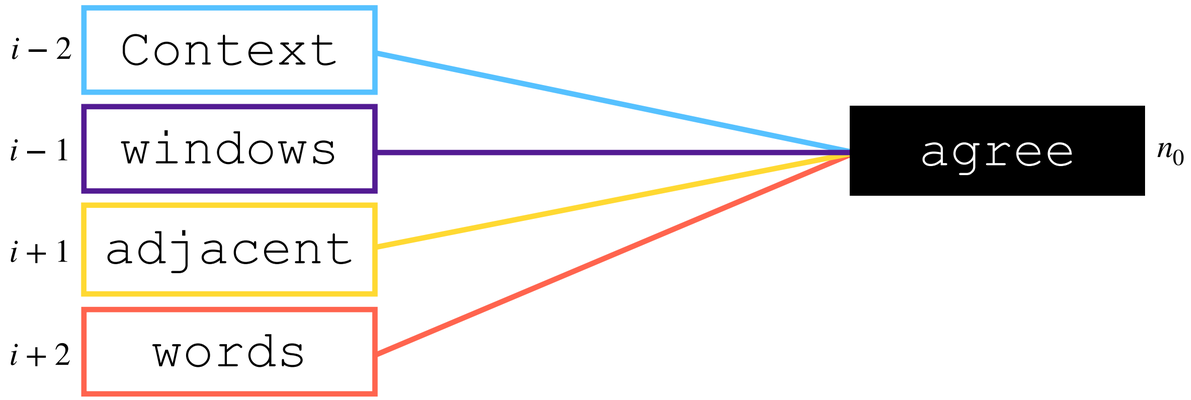
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**
- Opportunity: Operations have **independent order**
- Solution: Use registers for **maximum reusability**
  - Minimize up-front memory latency, maintain locality
  - Improved pipeline utilization
  - Maintain scheduling flexibility, reduce stress for Shared Memory



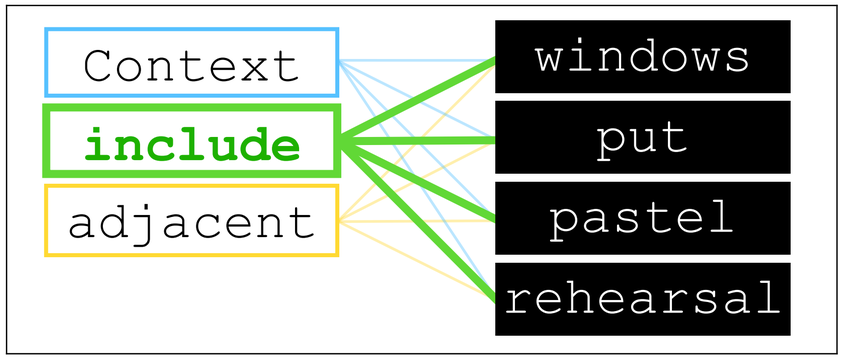
Negative Sample Reuse: Register-W2V

- Challenge: Negatives are **random** and have **lower reuse**
- Opportunity: Operations have **independent order**
- Solution: Use registers for **maximum reusability**
  - Minimize up-front memory latency, maintain locality
  - Improved pipeline utilization
  - Maintain scheduling flexibility, reduce stress for Shared Memory



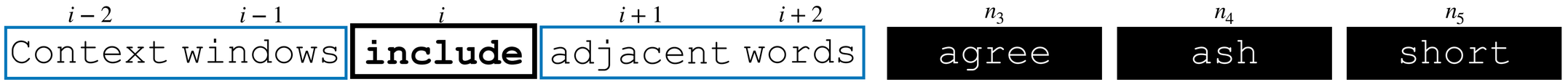
Context Word Reuse

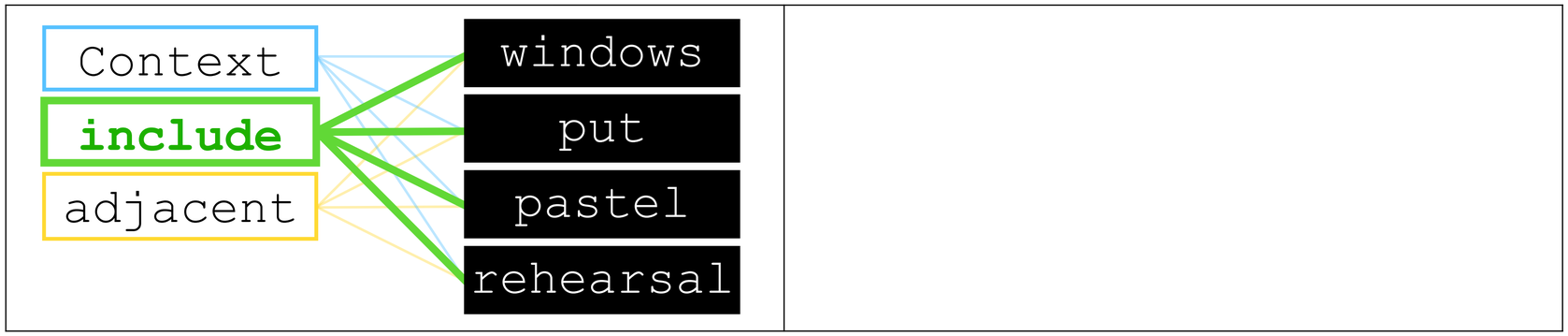
- Different Pattern: Context Words have **more reuse**



Context Word Reuse

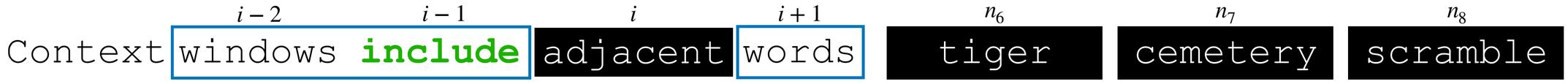
- Different Pattern: Context Words have **more reuse**

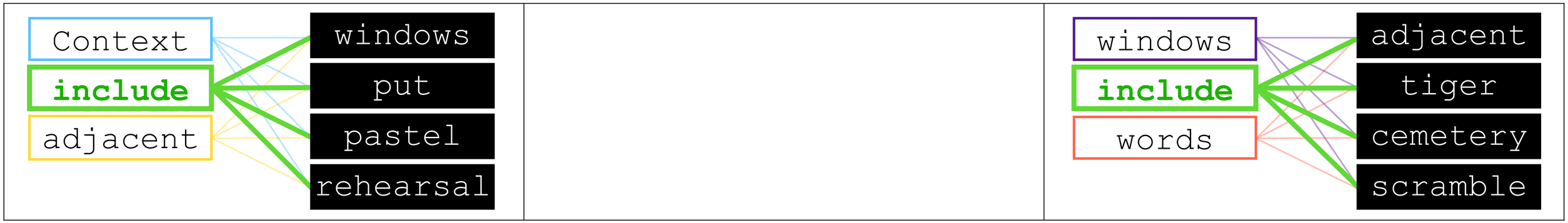




Context Word Reuse

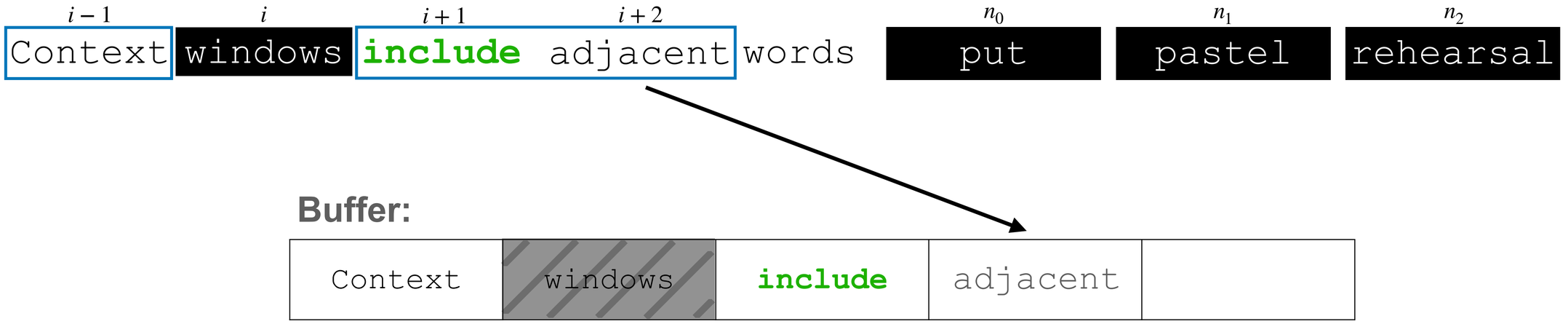
- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer





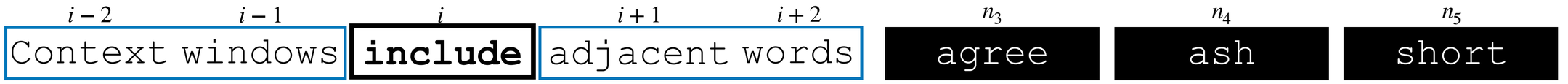
Context Word Reuse

- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer



Context Word Reuse

- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer

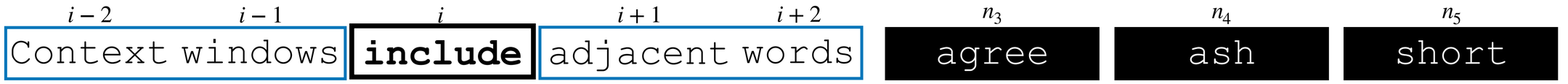


Buffer:

Context	windows	**include**	adjacent	words

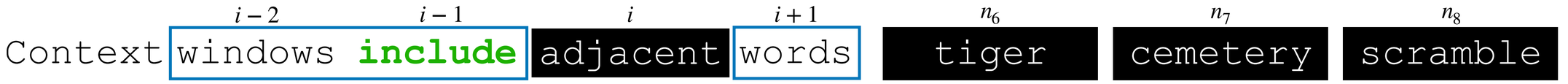
Context Word Reuse

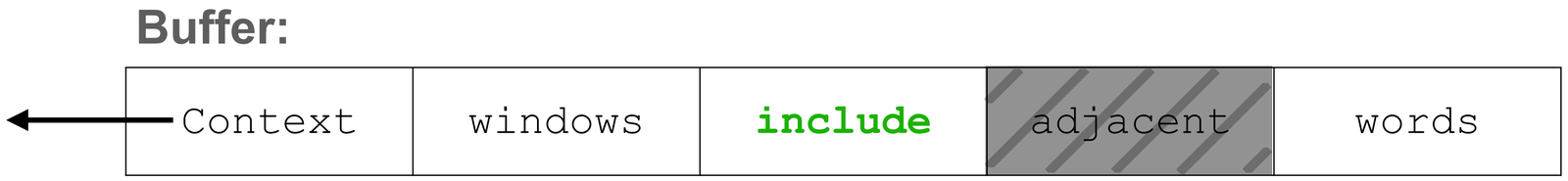
- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer



Context Word Reuse

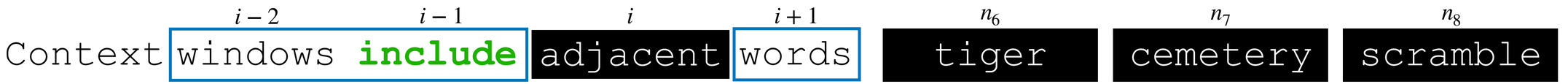
- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
  - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer





Context Word Reuse

- Different Pattern: Context Words have **more reuse**
- Allocation: Shared Memory leverages longer-term reuse
    - High performance; Explicit control; Flexible scheduling
- Management: Ring buffer



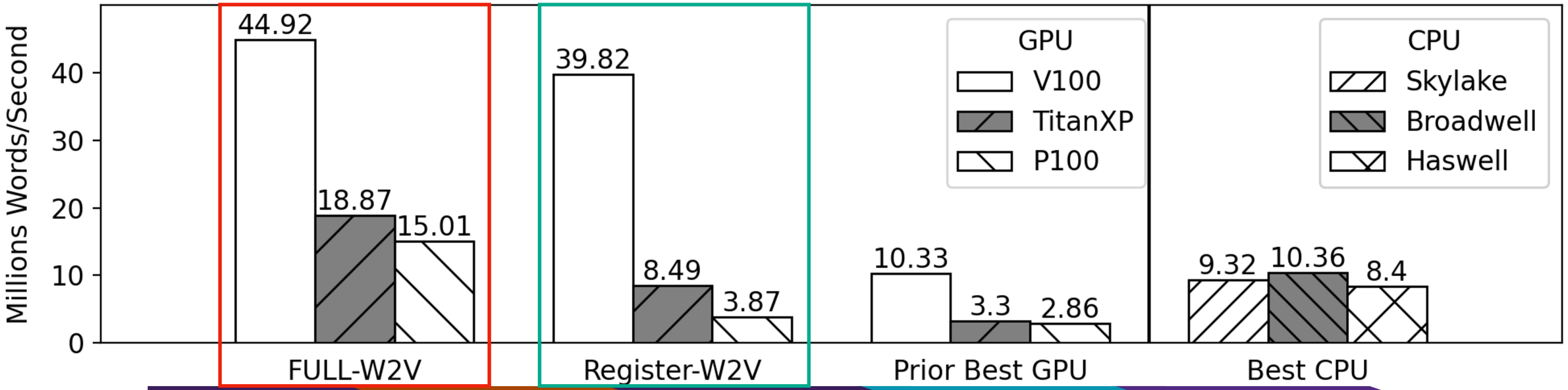
Buffer:

Results

Demand in GB/Epoch

Implementation	L1/TEX	L2	DRAM	%
Closest Prior	1,134.448	493.614	226.578	100.0%
Register-W2V	885.065	781.576	66.555	78.02%
FULL-W2V	**94.760**	**88.723**	**41.851**	**8.35%**

- FULL-W2V: Register-W2V + Context Word Reuse
  - **4.35X** total speedup previous best on V100
  - **3.85X** speedup from Register-W2V only
  - Sum data demand reduced by **91.65%**



Insights and Conclusion

- We present FULL-W2V
  - **4.35X** prior SOTA on V100
  - **2.99X** scaling from P100 to V100
- Different storage for different data
  - Register-W2V: maximize short term reuse in register
  - FULL-W2V: maximize long term reuse in shared memory
- Looking for more?
  - Our code is open source: https://github.com/tlranda/FULL-W2V
  - See the extended presentation for additional details

Acknowledgements

- Thomas Randall
    - tlranda@clemson.edu
    - tlranda.people.clemson.edu
    - https://www.researchgate.net/profile/Thomas-Randall-5
- Tyler Allen
    - tnallen@clemson.edu
    - tnallen.people.clemson.edu
    - https://www.researchgate.net/profile/Tyler-Allen-2
- Rong Ge
    - rge@clemson.edu
    - people.cs.clemson.edu/~rge/

- Support from NSF Grants CCF-1551511 and CNS-1551262
- Clemson University is acknowledged for generous allotment of compute time on Palmetto cluster