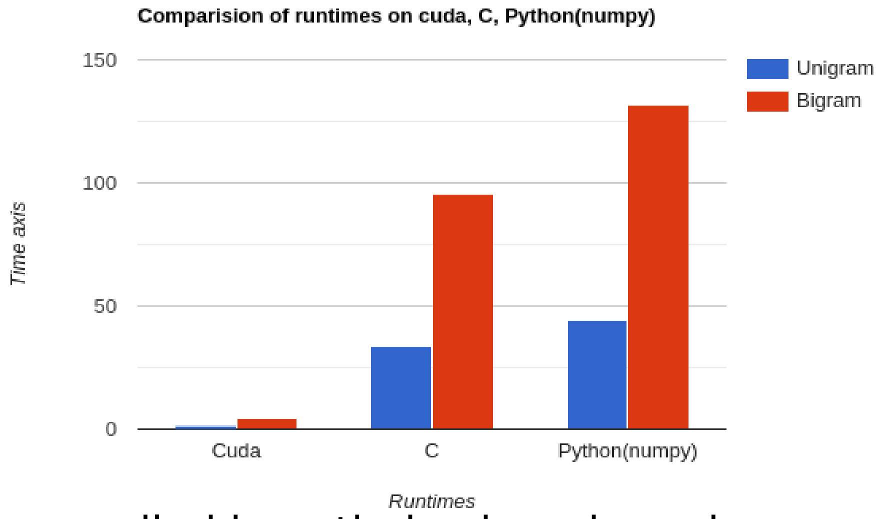
GPU Accelerated
Natural Language
Processing

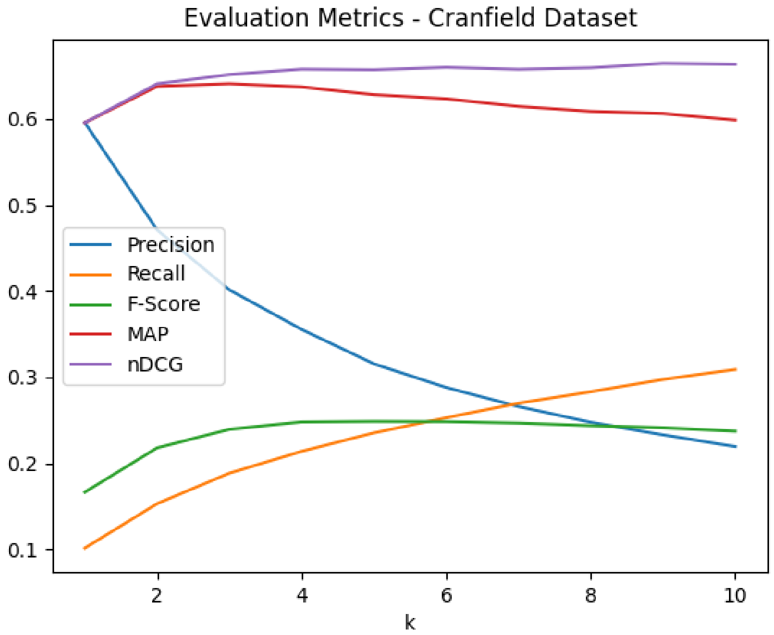
Optimised IR Search Engine on Unigram and Bigram models with comparison of parallel vs serial programming

Prof Rupesh Nasre
CH18B035 – Gunavardhan Reddy

Conclusions/Results



- Ranking of documents in cranfield dataset on 225 queries which took ~**1625s** on sequential python code now takes ~**1.47s** on cuda by performing **225*1400** parallel GPU operations, which is almost ~**1100** times faster
- Note the times shown in bar graph is of C functions compiled in optimised mode and python optimised with numpy libraries which use multithreading. And we notice an improvement of ~**40** times on C and ~**50** times on python



- Increased performance time on bigram model is due to vocabulary size of ~**25000** compared to that of ~**8300** on unigram model
- Comparision of models unigram and bigram using nDCG@(2) measure which takes into account relevance of the documents we notice an improvement from ~61% to ~64%
- ~**3**% improvement which is huge can be asserted to the fact that the documents in the cranfield dataset are **highly correlated and can be put under 5 to 10 topics** as the bigram model takes into consideration of collocations and cooccurrences of words.*

PROBLEMS FACED

- **Working with different languages, though python offers the flexibility of working with cuda and c, there are issues like:**
    - **Conversion of datatypes from python to cuda and c**
    - **Debugging erros in other languages as python does’t display the error**
    - **Memory allocation issues when transferring huge data**
- **Dataset isn’t big enough to notice the difference in GPU loading times on Unigram and Bigram models**

* **ref -** https://ieeexplore.ieee.org/document/7959986