RDFVault: A Compact In-Memory Dictionary For RDF data

Hamid R. Bazoobandi, Steven de Rooij, Jacopo Urbani, Annette ten Teije, Frank van Harmelen, and Henri Ba

Vrije Universiteit Amsterdam
University Van Amsterdam

3rd June, ESWC 2015

Dictionary

<http://xmlns.com/foaf/0.1/Person>	1
<http://purl.org/dc/terms/title>	2
<http://xmlns.com/foaf/0.1/name>	3

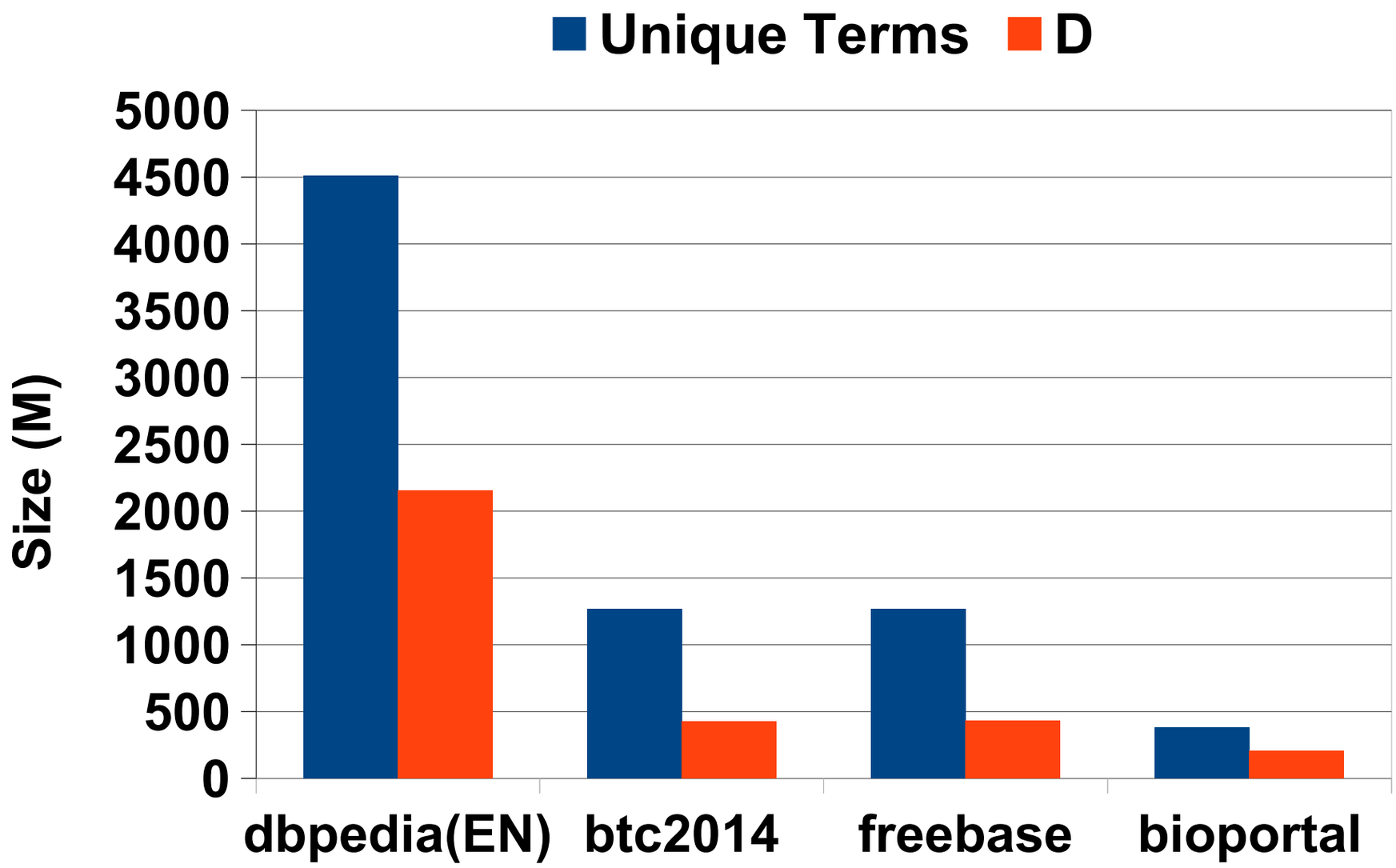
Encoding

1	<http://xmlns.com/foaf/0.1/Person>
2	<http://purl.org/dc/terms/title>
3	<http://xmlns.com/foaf/0.1/name>

Decoding

Dictionary Encoding is designed to maximize the data compaction, the dictionary itself is not compact

D of HDT


Unique Terms
D
Size (M)
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
dbpedia(EN)
btc2014
freebase
bioportal

Dictionary Encoding

- **For static data, encoding/decoding can be done at pre/post processing phases**
- **For dynamic/streaming data, applications have to maintain the dictionary in memory**

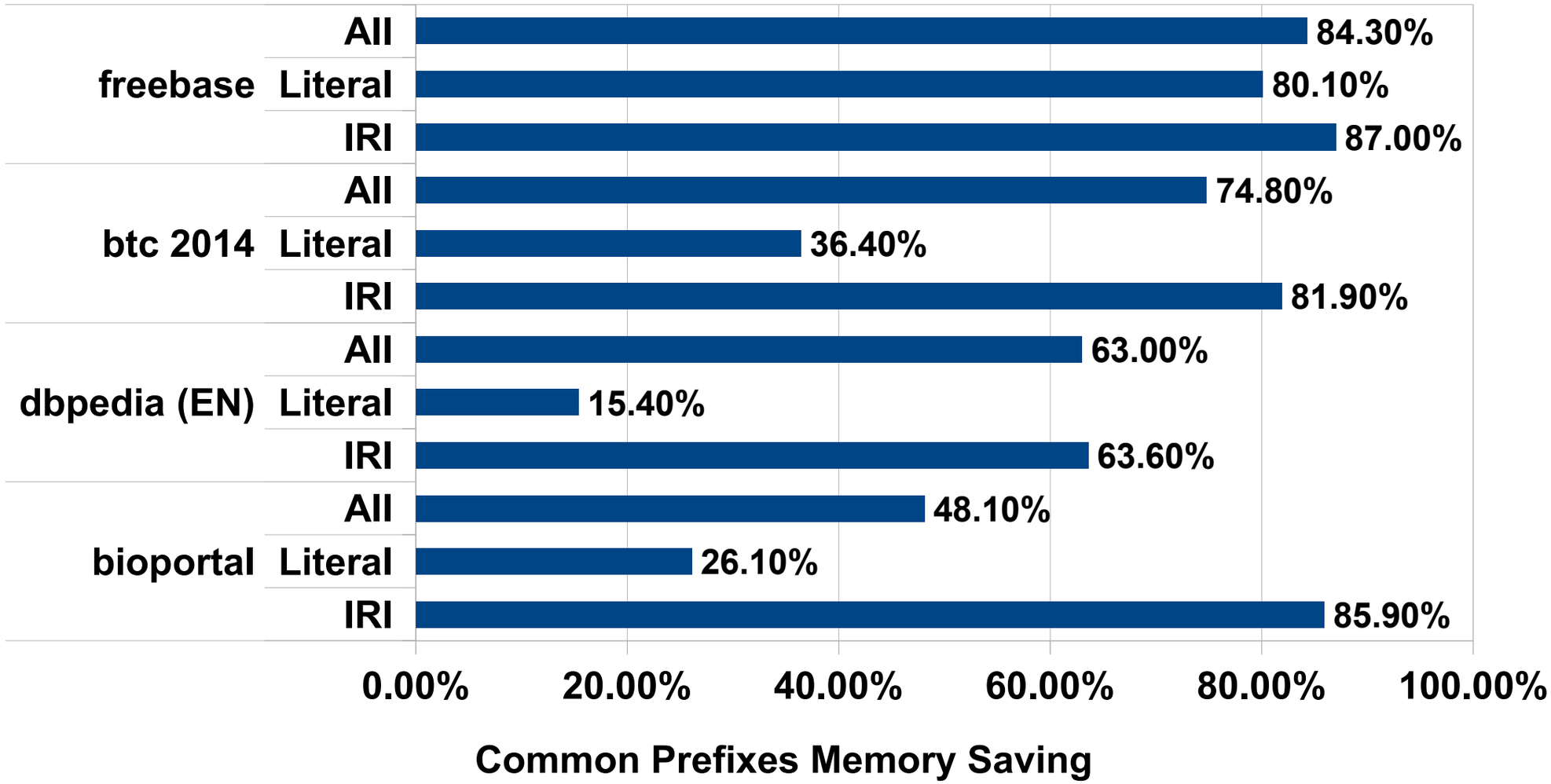
The Goal

- **A compact dictionary encoder for in-memory that supports frequent updates**

Method

- **Minimize the overhead of data structure**
- **Minimize the storage of common prefixes**

Common Prefixes Memory Consumption



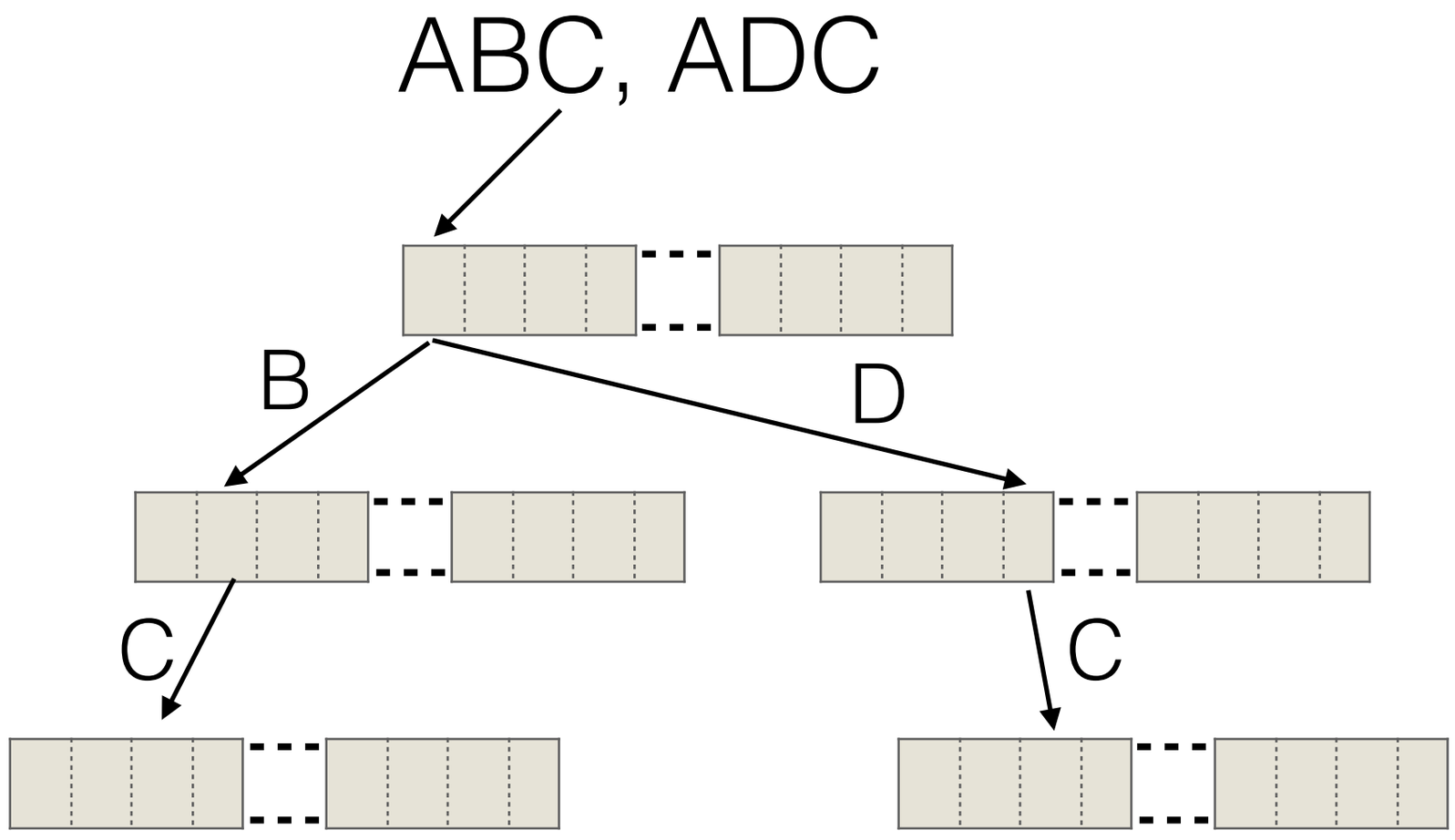
Trie Properties

- Keys are implicitly stored in the body of trie

- Keys can be reconstructed with *O(L)* complexity where *O(L)* is the length of string.

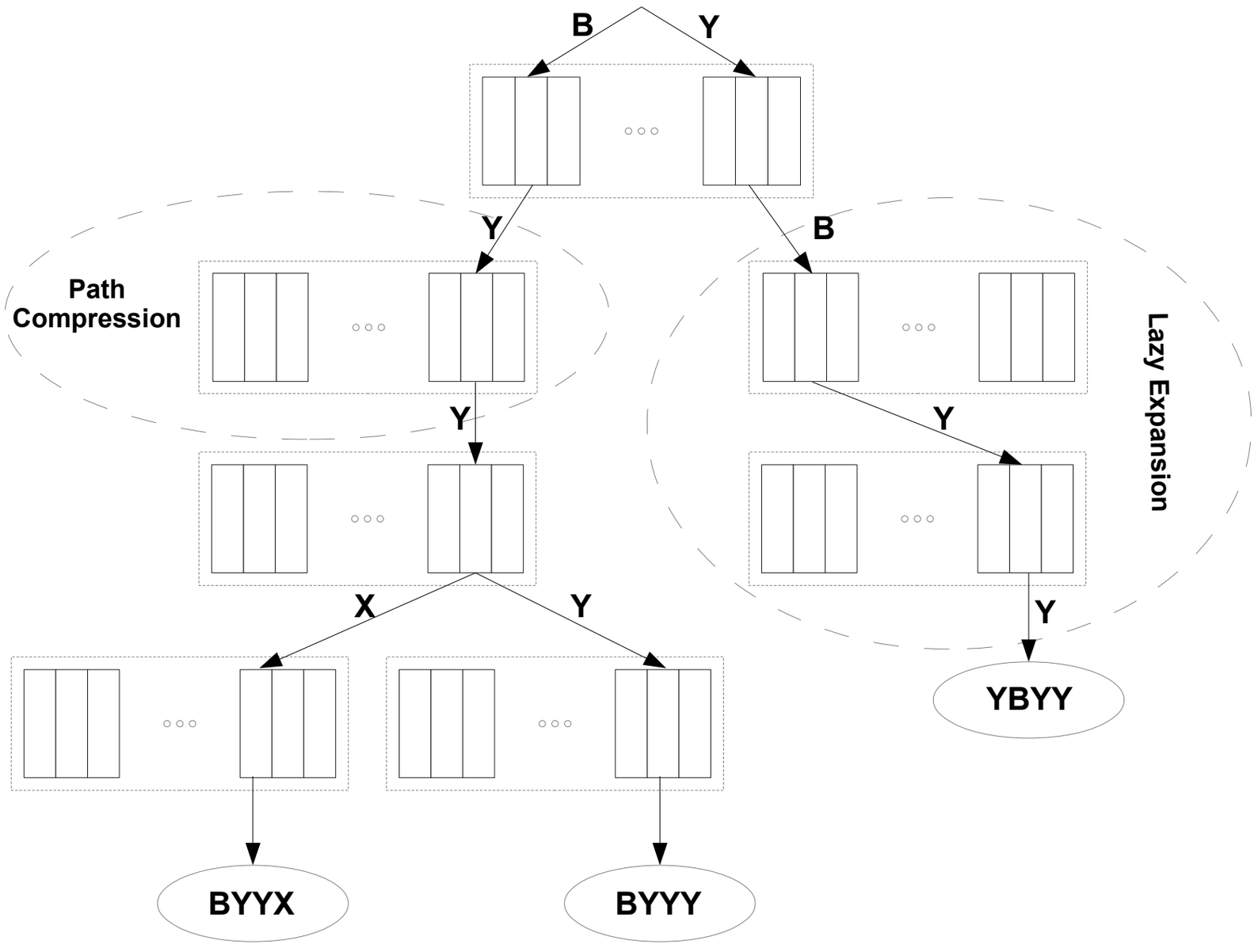
Trie Example

String Length



Extremely BAD memory efficiency

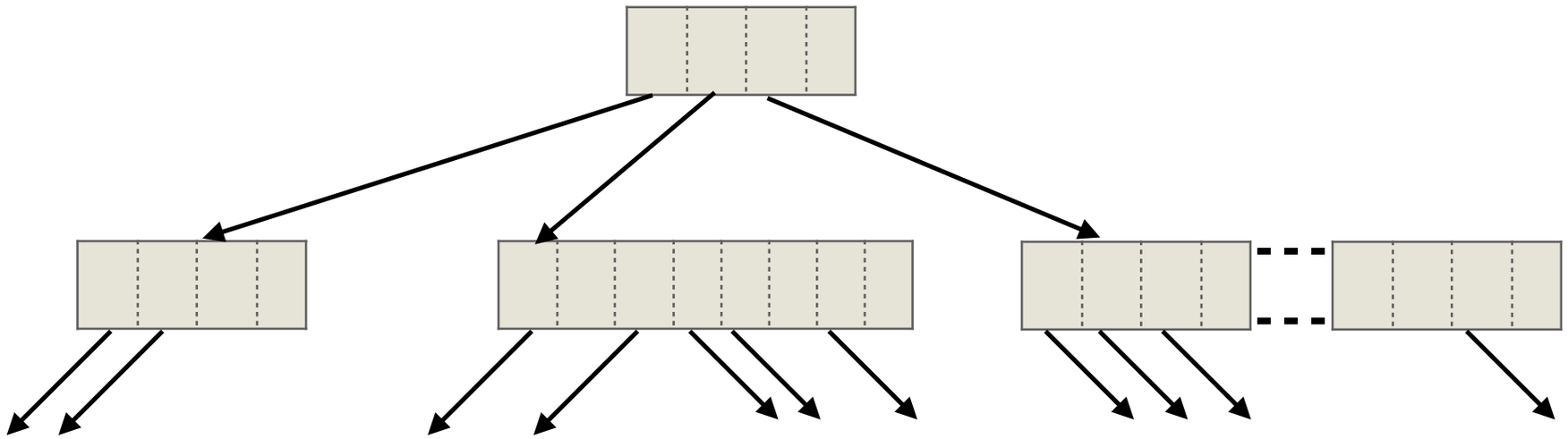
Compact Trie



Compact Trie Memory Efficiency

Dataset	Used Pointers
bioportal	1.58%
dbpedia (EN)	1.19%
freebase	1.91%
btc 2014	1.23%

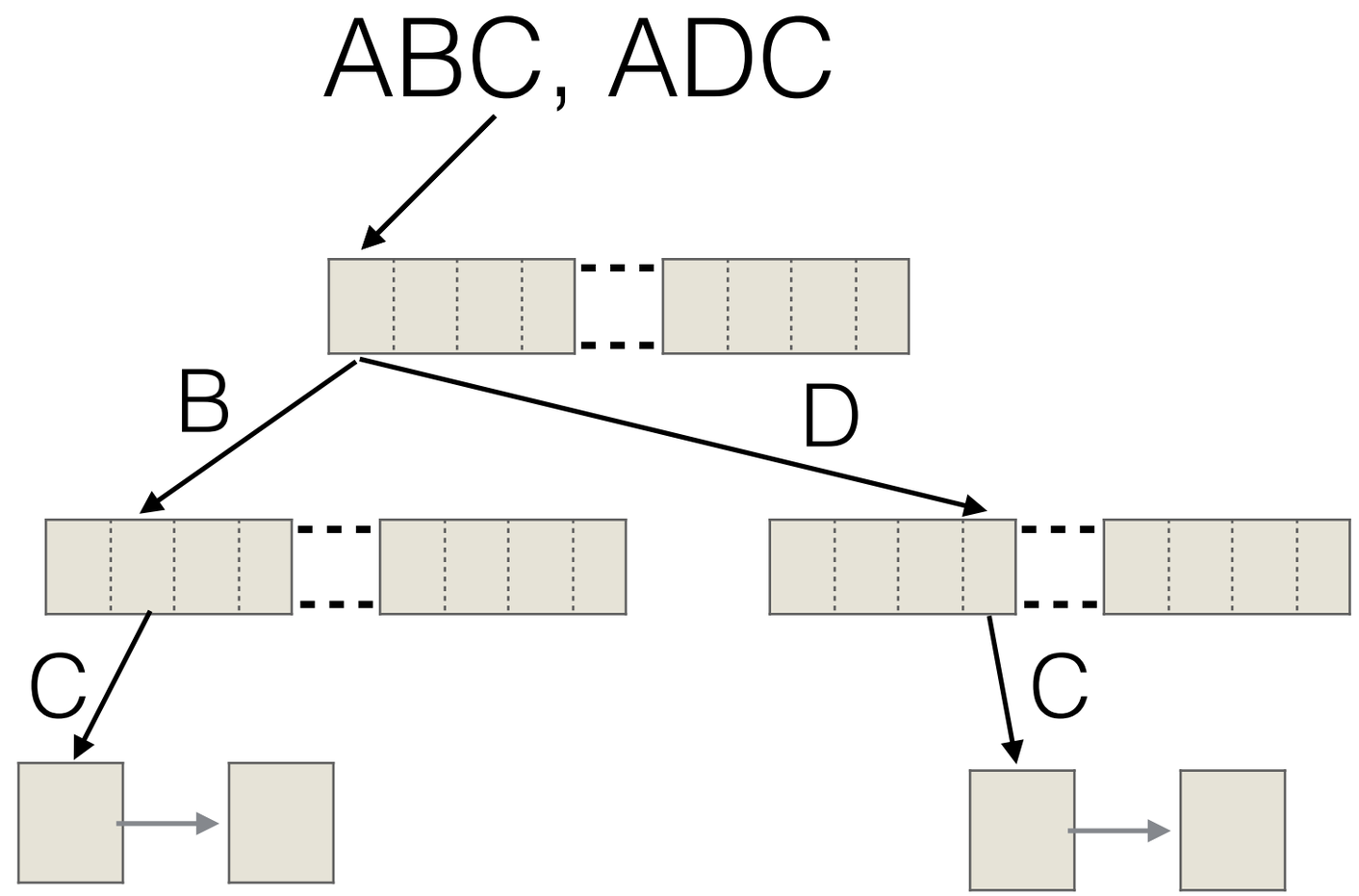
Adaptive Radix Tree (ART)



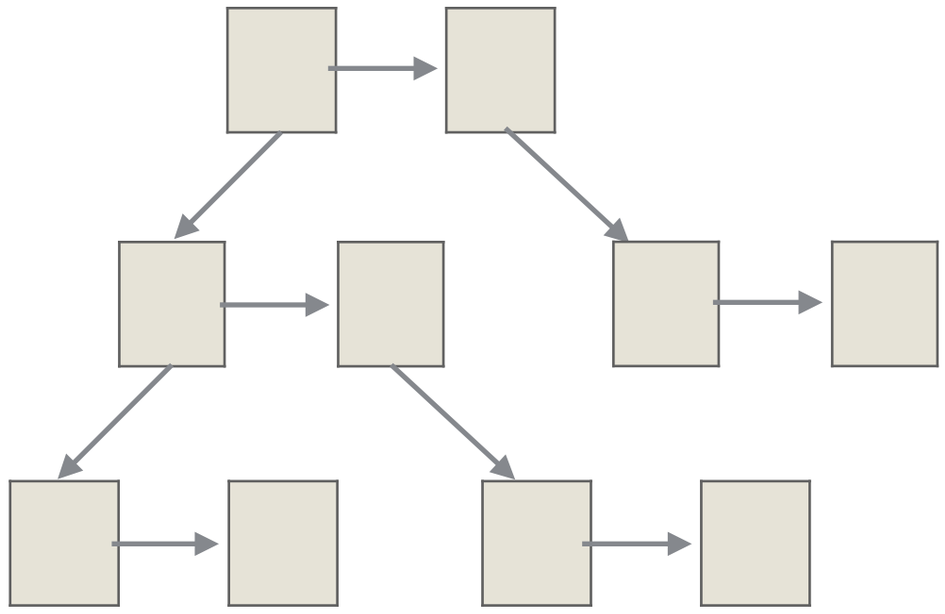
Adaptive Radix tree Memory Efficiency

Dataset	Used Pointers
bioportal	47.90%
dbpedia (EN)	46.60%
freebase	48.03%
btc 2014	44.79%

Burst Trie & HAT

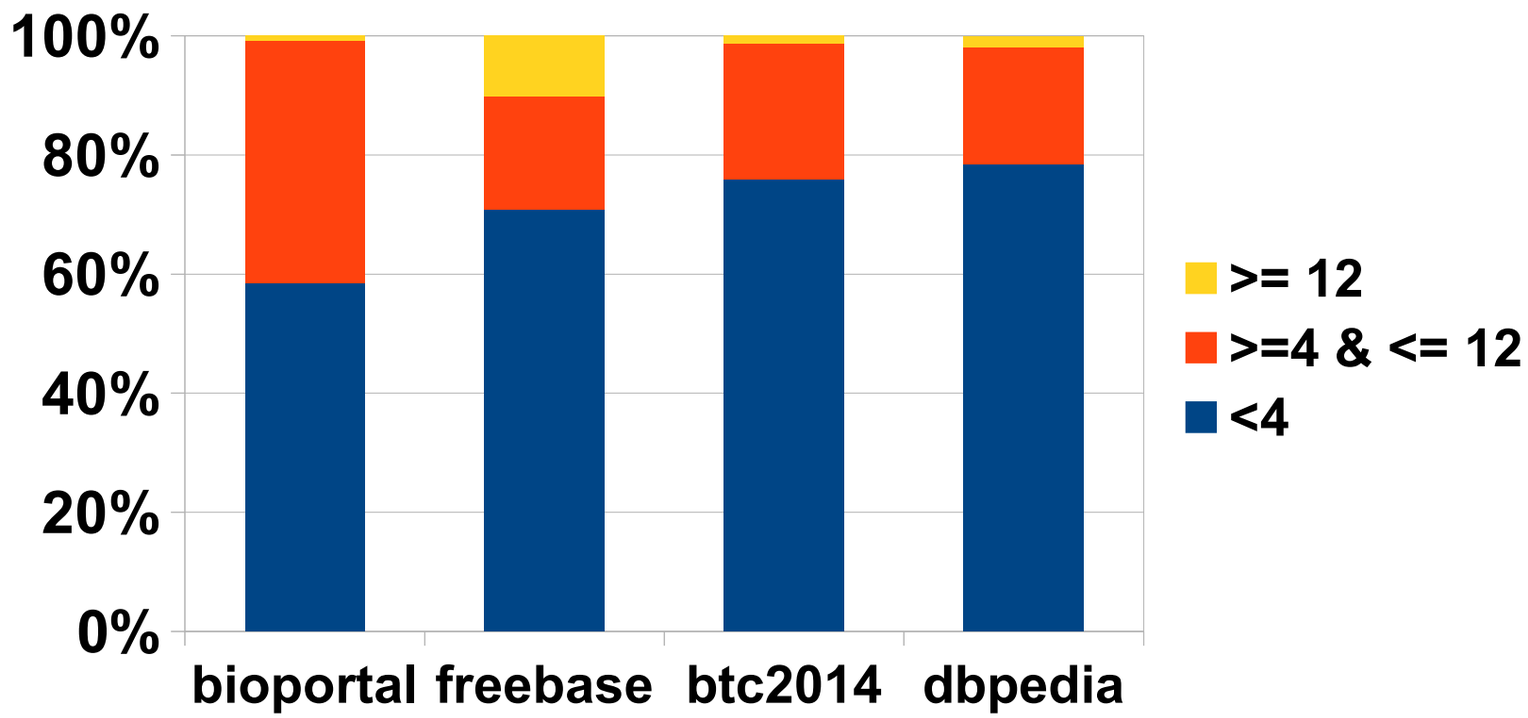


List Trie



- Very memory efficient
- Very slow for generic data
- **NO ONE USES THIS**

Children distribution

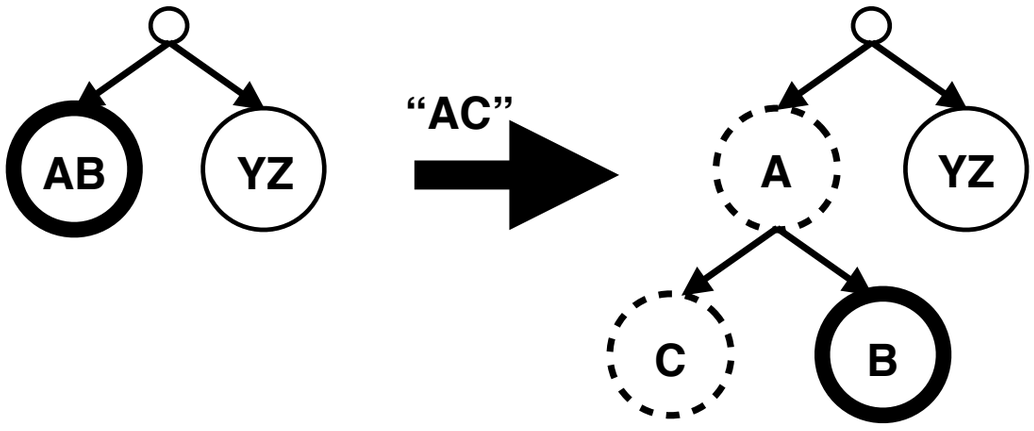


- **For majority of nodes, the cost of using linked lists is not significant**
- **RDF data is highly skewed, therefore some paths in the trie are traversed much more frequently**

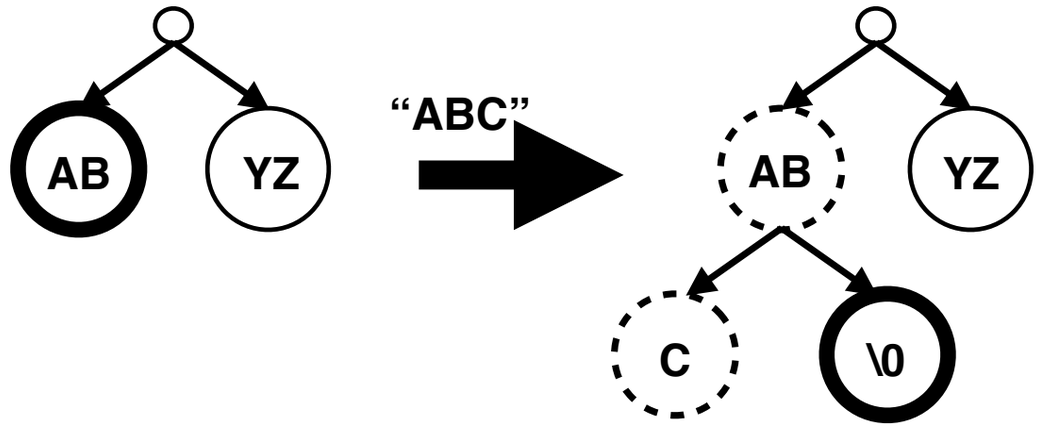
Trie used in RDFVault

Compact List Trie, with Move-to-front Policy

Preserving Pointers

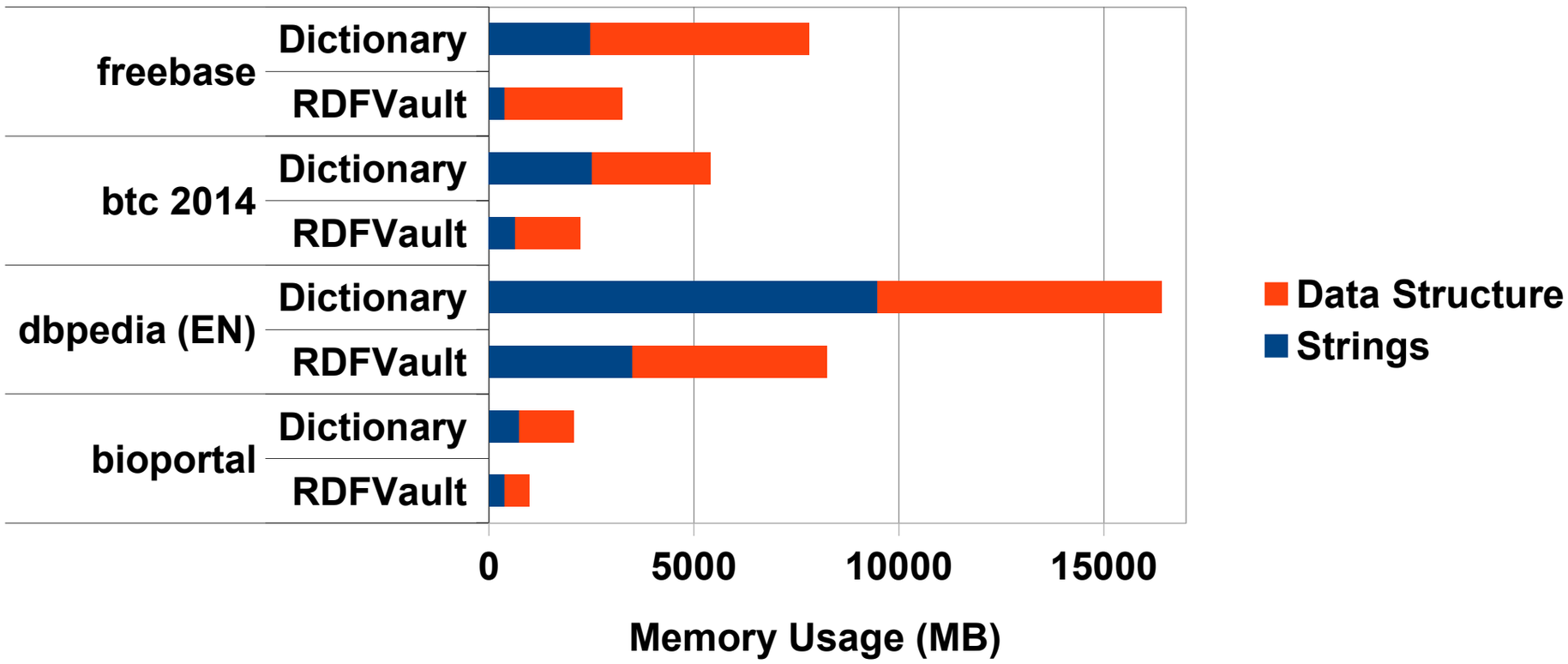


(a)

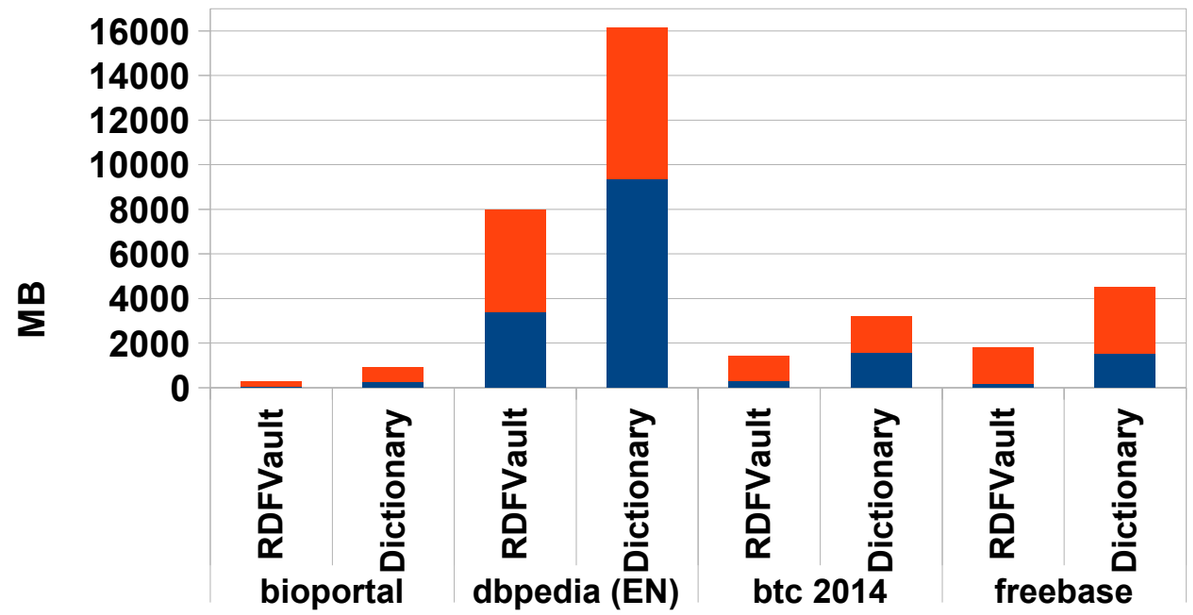


(b)

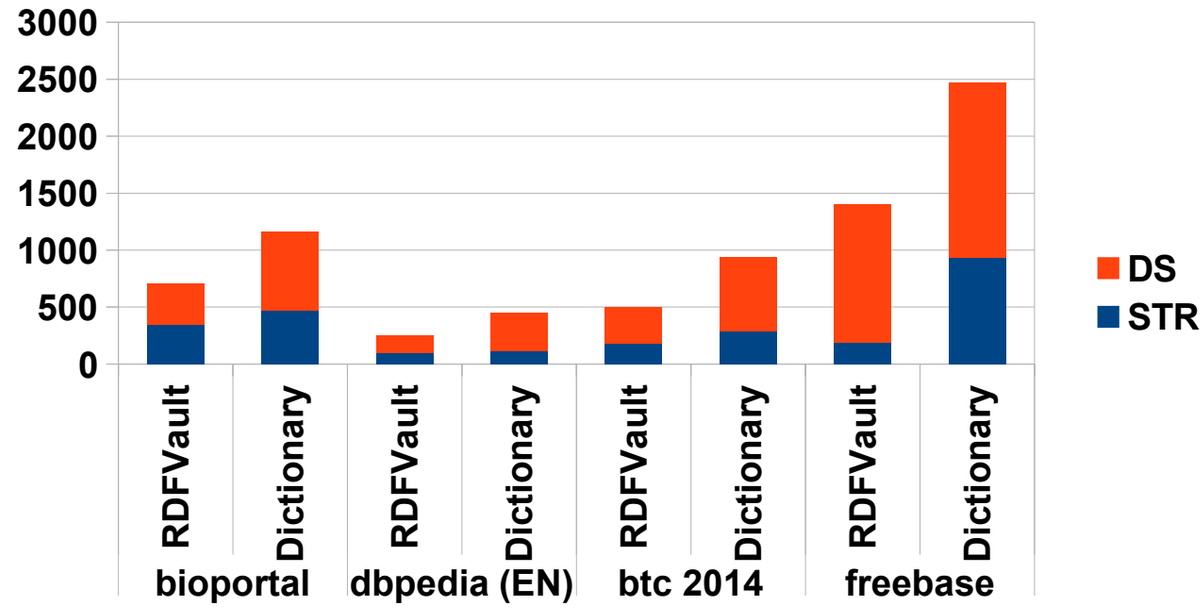
Memory Efficiency



Memory Efficiency

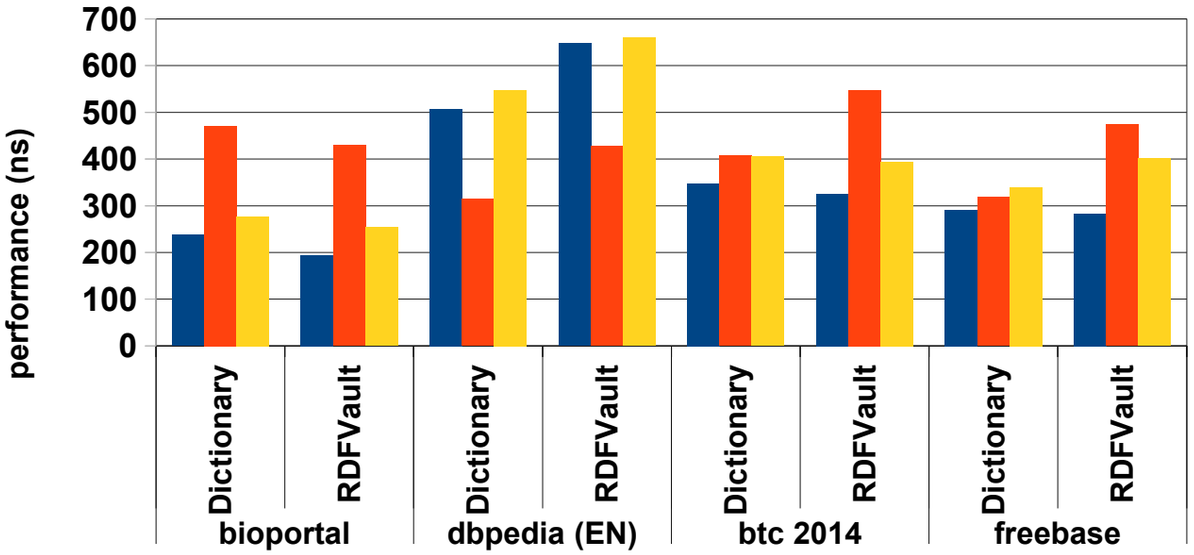


(IRIs)

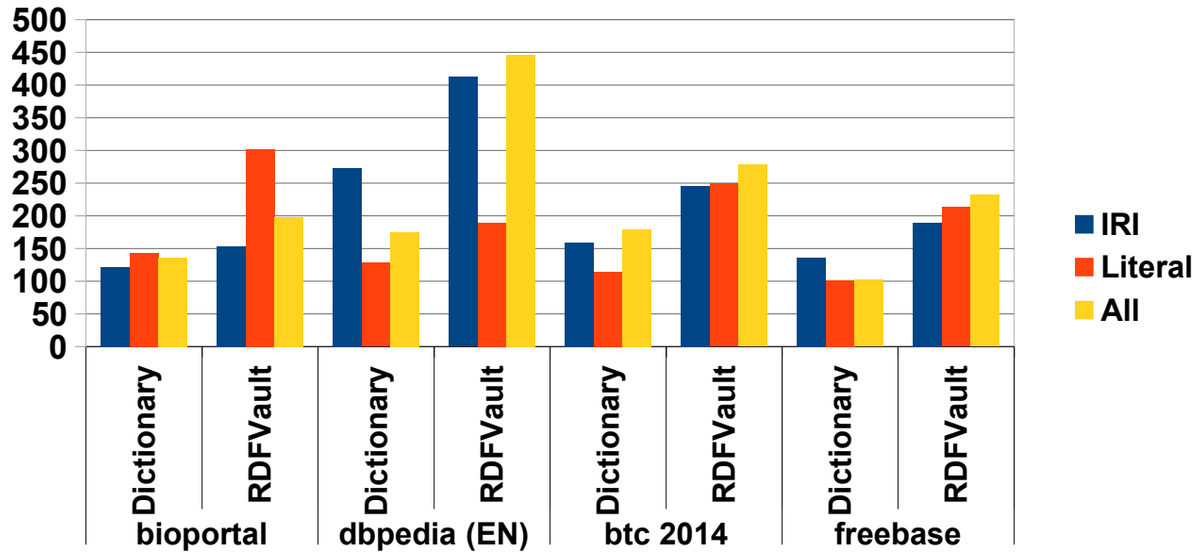


(Literals)

Performance

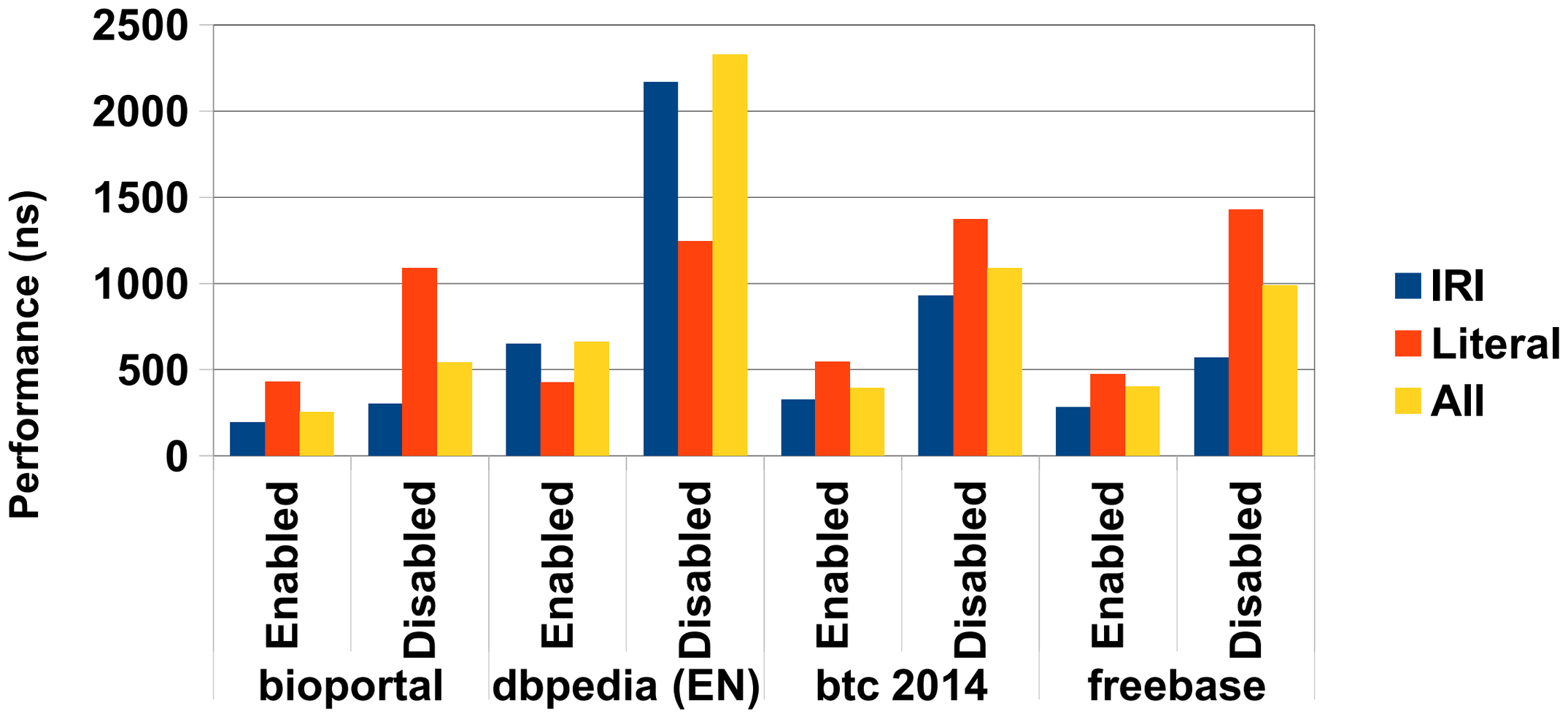


(Encode)



(Decode)

Move-to-front Effectiveness



Conclusion

https://github.com/bazoohr/RDFVault.git