Robust Cache System for Web Search Engine Yioop

Presented by -

Rushikesh Padia

Department of Computer Science

SJSU SAN JOSÉ STATE UNIVERSITY

Committee:

Dr. Chris Pollett (Advisor)

Dr. Ben Reed

Dr. Robert Chun

Every millisecond counts!

Agenda

- Introduction
- Preliminary Work
- Implementation Details
- Results
- Conclusion
- Future work

Introduction

- Yioop is an open source web search engine
- It uses result cache to improve response time
- Current implementation uses dynamic cache based on Marker Algorithm
- A dynamic cache based on Marker or LRU algorithm captures short-term trends
- The goal of the project is to explore different caching strategies and implement them in Yioop

Yioop Search Engine Architecture

- Yioop search engine consists of three main components: Crawler, Indexer, and Query Processor.
- **Crawler**
  - Responsible for discovering and gathering information from web pages.
  - **QueueServer** process queues URLs to be fetched.
  - **Fetcher** process fetches webpages from the internet.
- **Indexer**
  - Processes fetched pages to extract textual content.
  - Builds an inverted index, which aids in processing queries.
  - Generates textual summaries of the extracted information.

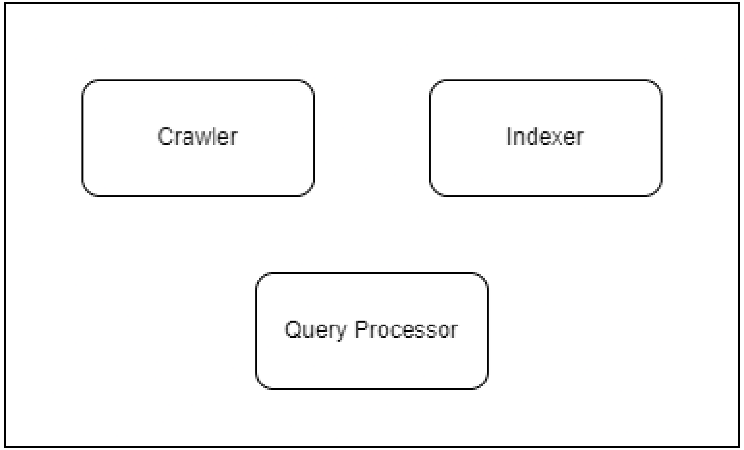


Figure 1: Yioop Architecture

SJSU SAN JOSÉ STATE UNIVERSITY

Yioop Search Engine Architecture

- **Query Processor**
  - Evaluates user queries for search results.
  - Cleaning and preprocessing the query through techniques like case folding, stemming, and stopword elimination.
  - Utilizes a result cache to check if the results for a query are already computed.
  - If results are available in the cache, they are returned without further processing, saving time and resources.
  - If not available, the processor retrieves posting lists for each term in the query from the inverted index.
  - Utilizes a ranking algorithm to determine the most relevant documents.

Preliminary Work

- To implement new cache system in Yioop, following algorithms were evaluated
  - Static-Dynamic Cache
  - Machine Learning Static-Dynamic Cache
  - Static-Semistatic-Dynamic Cache
  - Static-Topic-Dynamic Cache
- Experiments were performed to evaluate each of these algorithms
- Following slides give more information about each of these algorithms

Static-Dynamic Cache

- Dynamic cache adapts well to the short-term trend in queries
- It does not adapt well in presence of both short-term and long-term trends in queries
- Static-Dynamic cache divides cache into two segments
  - Static segment adapts to the long-term trends in the queries
  - Dynamic segment adapts to the short-term trends in the queries
- Static Cache - Modeled as offline cache allocation problem
- Dynamic Cache - Modeled as online admission-eviction problem
- Query is first checked in static cache and if not present, it is checked against the dynamic cache

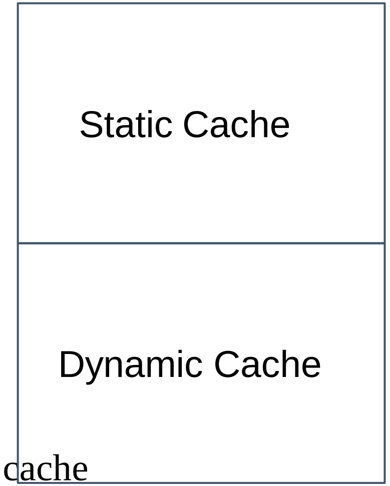


Figure 2: Static-Dynamic Cache

Machine Learning Static-Dynamic Cache

- Dynamic cache implemented using classical machine learning models.
- Goal of machine learning models is to accurately predict the next appearance of the query i.e. “IAT_NEXT”.
- Features extracted from the query.
- Models cache admission problem as a classification problem to classify whether query should be admitted or not.
- Models cache eviction problem as a regression problem to predict the “IAT_NEXT” value. It removes value having highest “IAT_NEXT”.

Dataset

- AOL Query logs contains 3 months of query logs generated in year 2006 on the AOL search engine.
- It contains total 36 million queries
- Dataset contains raw queries, anonymized user ids, timestamp, url clicked by the user, and the rank of the item.

	AnonID	Query	QueryTime	ItemRank	ClickURL
0	53	mapquest	2006-03-01 15:18:21	1.0	http://www.mapquest.com
1	66	cajun candle	2006-03-01 13:20:18	1.0	http://www.cajuncandles.com
2	66	candle jars	2006-03-01 13:22:29	1.0	http://www.sks-bottle.com
3	66	muic.com	2006-03-01 22:42:05	NaN	NaN
4	66	i need a company name	2006-03-02 12:32:59	NaN	NaN

Table 1: AOL Query log Dataset

Query Features

- Based on the available data, a subset of features from original paper were selected

Feature	Description
QUERY_HOUR	Hour of the day, the query was fired
LAST_MIN_FREQ	Frequency of query in last minute
LAST_HOUR_FREQ	Frequency of query in last hour
LAST_DAY_FREQ	Frequency of query in last day
PAST_FREQ	Total frequency of the query
IS_TIME_COMPAT	Whether query is time compatible
QUERY_LENGTH	Number of words in the query

Table 2: Query features

Experiment and results

- Regression model was fitted on the dataset of 1M queries
- Training data was highly skewed, over 80% data had IAT_NEXT value less 1% of total data
- Log binning was applied to remove the skewness of the data
- The model achieved the hit rate of 28.33% for cache size of 100 frames and 4K total number of queries
- LRU and Optimal offline algorithm achieved 52.6% and 57.27% for the same data.
- Will requires lot of tuning and feature engineering to achieve acceptable results

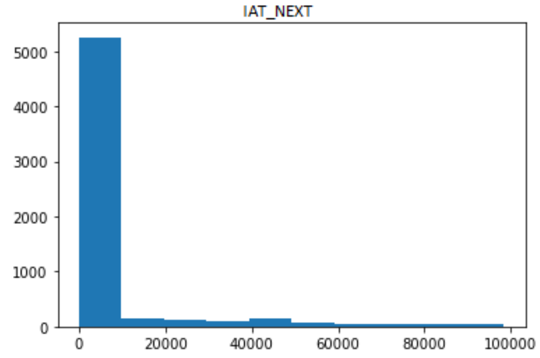


Figure 4:IAT_NEXT value before log binning

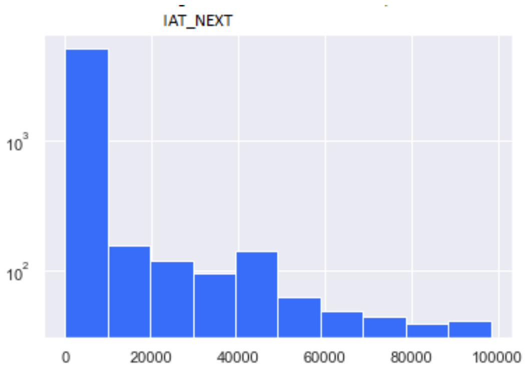


Figure 4: IAT_NECT value after log binning

Static-Semistatic-Dynamic Cache

- Adds Semistatic layer to Static-Dynamic cache framework
- Based on observations, day time popular queries are different than night time popular queries.
- Categorizes queries into day-time popular, night-time popular, and all-time popular queries
- Static segment contains all-time popular queries
- Semi-static segment toggles between day-time and night-time popular queries
- Dynamic cache implemented with LRU caching algorithm
- Query is checked in each cache in order - static cache, semi-static cache, and dynamic cache

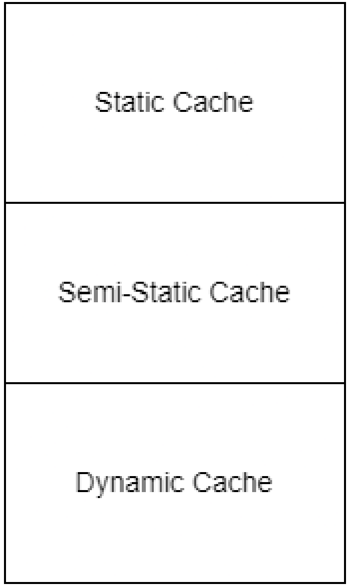


Figure 5: Static-Semistatic-Dynamic cache

SJSU SAN JOSÉ STATE UNIVERSITY

Experiment and Results

- Query categorization was done using AOL query logs
- Query which appeared more than 80% of time in day-time were categorized was day-time and similarly for night-time queries
- If night day-time or night-time, it was termed all-time query
- Training was performed using 1.5M queries
- Results were evaluated on other set of 1.5 M queries with cache size of 300 frames.

Experiment and Results

Cache Size	Configuration	Hit Rate
300	SDC(80-20)	56.65%
300	SSDC(10-70-20)	55.76%
300	SSDC(20-60-20)	55.95%
300	SSDC(40-40-20)	56.21%
300	SSDC(60-20-20)	55.94%
300	SSDC(70-10-20)	56.56%
300	SSDC(0-80-20)	55.46%

Table 3: Hit rate in percentages (%) for 1.5M queries and cache size of 300 entries

- The algorithm has acceptable performance. Only 1.2% lower than Static-Dynamic cache.
- There was no scope for improvement in this approach
- Requires large cache space and as day-time night-time both requires persistence, it was not selected.

Static-Topic-Dynamic Cache

- Different topics for e.g. weather, tv shows are accessed more frequently during different time of days and have different access patterns
- Static-Dynamic cache does not adapt well to these type of access patterns
- Static-Topic-Dynamic cache adds Topical layer over Static-Dynamic cache to capture such trends
- Topic is assigned to queries using some topic model
- Each topic has own instance of cache managed by a certain policy
- Query is checked in each cache in order - static cache, topical cache, and dynamic cache

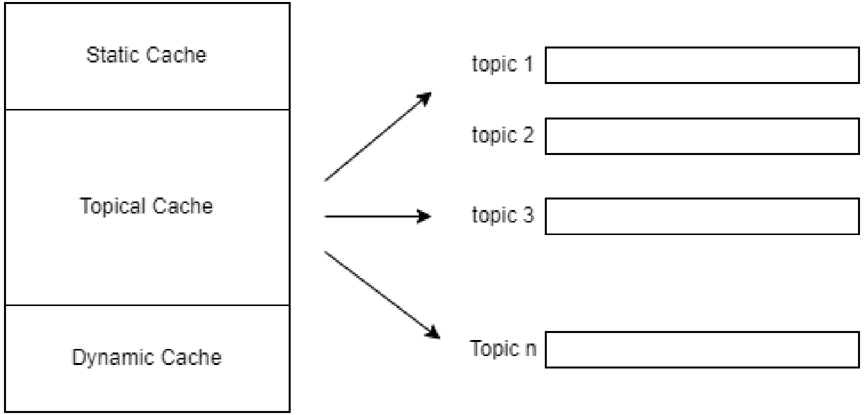


Figure 6: Static-Topic-Dynamic Cache

Topic Modeling

- Topic Modeling is a popular technique in NLP to extract topics from text
- Extracts latent topics without unsupervised learning algorithms
- Popular algorithms
  - Latent Semantic Analysis (LSA)
  - Latent Dirichlet Allocation (LDA)
  - k-means Algorithm

Experiments and Results

- Experiment was performed on 10K queries and cache size of 100 and 200 frames
- For topic modeling LDA model was used
- LDA model was trained using 1.2M news headline dataset
- Each topical cache instance was governed using LRU cache
- Dynamic cache was implemented using LRU cache

Experiment and Results

- Static-Dynamic cache performance was close to Static-Dynamic cache
- There was scope for improvement by training the model using actual search engine text data
- Queries can be enriched using user's clicked URL data

Cache Size	Configuration	Hit Rate
100	LRU	42.33%
100	Belady	49.08%
100	SDC(30-70)	43.35%
100	STDC(30-50-20)	43.16%
100	STDC-V(30-50-20)	43.10%
200	LRU	44.49%
200	Belady	49.81%
200	SDC(15-85)	45.16%
200	SDTC(15-50-35)	45.08%
200	STDC-V(15-50-35)	45.01%

Table 4: Hit rate in percentages (%) for 10K queries and cache size of 100 and 200 entries

Implementation of New Cache in Yioop

- Choice of caching algorithm depends on the use case
- Yioop is used of variety of purposes for e.g. general purpose crawling, crawling set of web pages or crawling user's website
- Old caching system in Yioop was tightly coupled with Marker Algorithm
- The new system gives Yioop flexibility to switch between different cache types
- Following cache type are added in Yioop [Demo 1]
  - Least Recently Used
  - Static-Dynamic Cache
  - Static-Topic-Dynamic Cache

Cache System Design

- Object Oriented Design of new cache system
- Flexible design to switch internal implementation of Static-Dynamic and Static-Dynamic cache

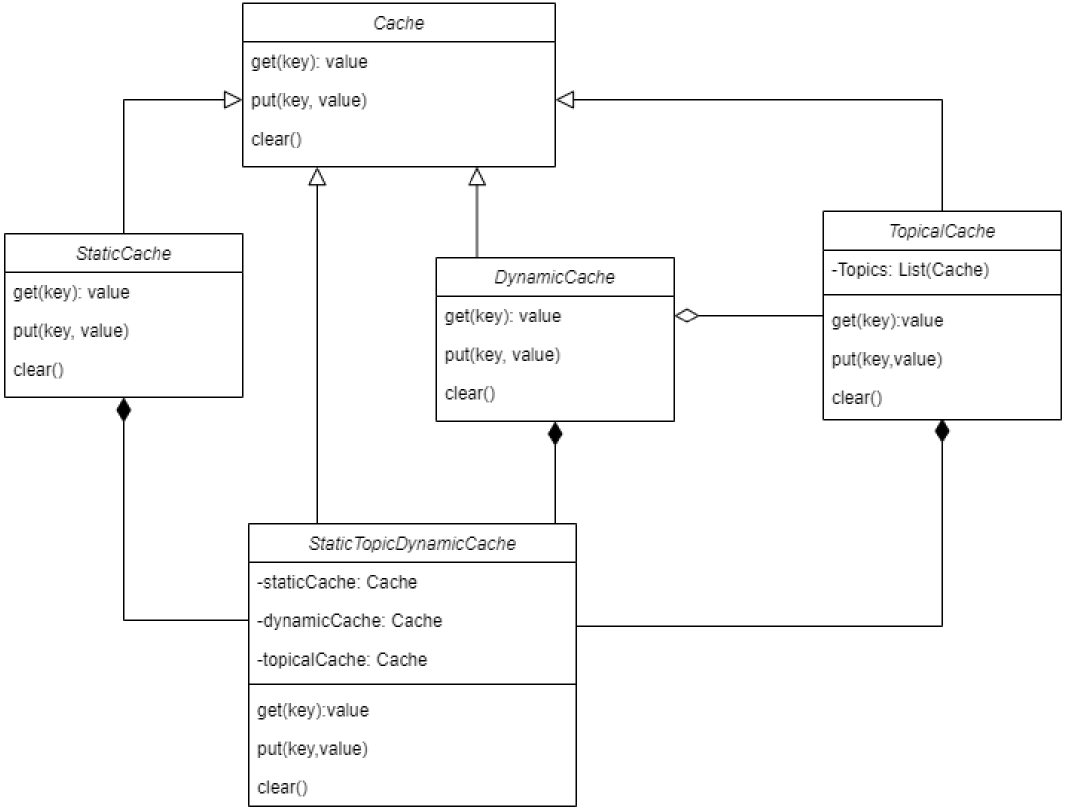


Figure 7: Class diagram of Cache System in Yioop

Implementation Static-Dynamic Cache

- "StaticDynamicCache" class is implemented in Yioop
- Static cache is populated using most frequent queries in search engine logs
- Dynamic Cache segment uses instance of LRU cache

Implementation of Topical Cache

- "TopicalCache" class is implemented in Yioop
- It uses k-means clustering topic model to extract topic from cache
- k-means algorithm is an unsupervised machine learning model used of clustering
- k-means algorithm can also be used as topic model where centroids of k-means acts as latent topics
- k-means algorithm can be trained to classify text into k number of topics
- Each of these topic have corresponding cache governed by "LRUCache"

Dataset

- To train k-means algorithm a text data was created using Yioop's indexer
- As queries are usually 2-3 words long, it needs to be enriched with additional contextual information
- Contextual information was added using user's clicked URL and webpages extracted from Yioop's crawl data
- Yioop summarizer's text was added to the query to enrich the queries for training
- As Yioop does not get lot of traffic, instead of Yioop's query logs, AOL query logs were used
- Total of 10K clicked URL queries were used for dataset creation
- Thus a dataset was created using combination of Yioop's crawl data and AOL query logs

Word Embeddings

- Machine learning algorithms requires text to be represented as vectors
- In Yioop, CountVectorizer is implemented to convert terms into vectors
- CountVectorizer first creates vocabulary from text corpus
- Assigns each word a unique index in a vector
- Increments count of index of each term in the text
- To reduce the cost of memory and cpu, all vectors are implemented as sparse vectors in Yioop

Training k-means algorithm

- To train k-means algorithm, each document in the training dataset was converted into the vector
- Each vector was appended to form a document-term matrix
- k=10 centroids were selected to train the algorithm
- To avoid training and creating vocabulary, serialization and deserialization capability is added to both KMeansClustering model as well as CountVectorizer

Results of k-means algorithm

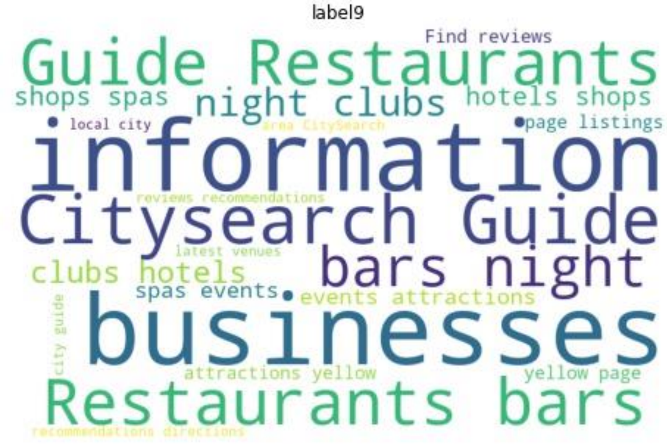


Figure 8: Restaurants

Figure 9: Holidays

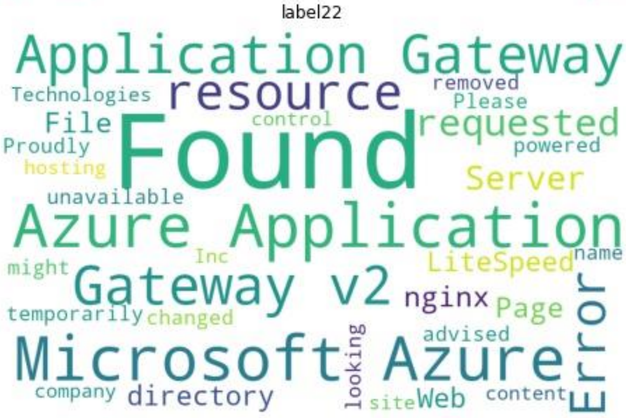


Figure 10: Technology

SJSU SAN JOSÉ STATE UNIVERSITY

Implementation of Static-Topic-Dynamic Cache

- “StaticTopicDynamicCache” is implemented in Yioop
- Uses “TopicalCache” for topical segment of it’s cache
- Static and dynamic segments are implemented with “StaticCache” and “LRUCache”
- “StaticTopicDynamicCache” first checks whether cache is present in static cache.
- If it is not present it checks in topical Cache.
- If it is not present it checks with dynamic cache.
- If it is found in any segment, result hit is returned

Evaluation of Different Cache Types in Yioop

- To evaluate the performance of cache, “CacheMetricWrapper” class is implemented
- It delegates get and put calls to the internal cache implementation and tracks performance based on the output
- Currently it tracks numbers of hits and misses and hit rate
- The performance was evaluated on 5k, 10k, and 20k queries
- Cache sizes were varied between 200-500 entries

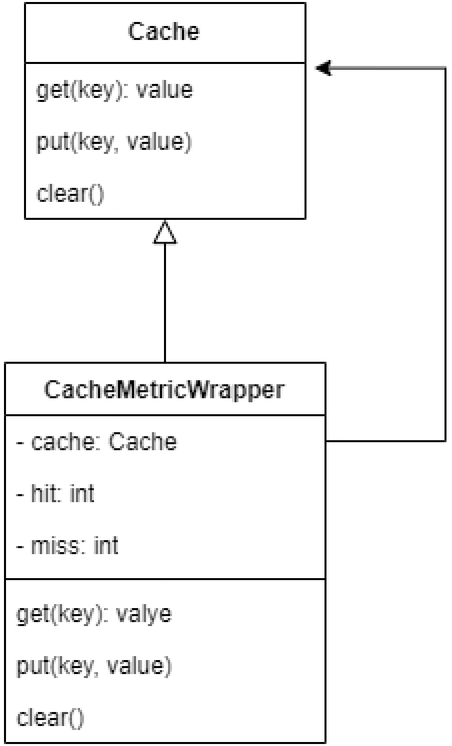


Figure 11: CacheMetricWrapper class diagram

Results

Cache Size	LRU	SDC (20-80)	SDC (30-70)	SDC (40-60)
200	69.8	70.6	71.04	71.4
300	69.96	71.12	71.72	72.34
400	69.98	71.58	72.4	73.2
500	70	72.02	73.02	74.04
Cache Size	STDC (20-40-40)	STDC (20-50-30)	STDC (20-60-20)	STDC (20-70-10)
200	70.22	69.96	69.68	69.56
300	70.98	70.74	70.54	70.36
400	71.46	71.38	71.12	70.9
500	71.92	71.88	71.72	71.56

Table 5: Hit rate in percentages (%) for 5K queries and cache size between 200-500 entries

Results

Cache Size	LRU	SDC (20-80)	SDC (30-70)	SDC (40-60)
200	61.38	61.96	62.21	62.31
300	61.55	62.3	62.6	62.91
400	61.6	62.54	62.98	63.41
500	61.65	62.79	63.36	63.87
Cache Size	STDC (20-40-40)	STDC (20-50-30)	STDC (20-60-20)	STDC (20-70-10)
200	61.54	61.26	60.95	60.73
300	62.1	61.89	61.53	61.33
400	62.42	62.29	61.99	61.69
500	62.7	62.66	62.44	62.2

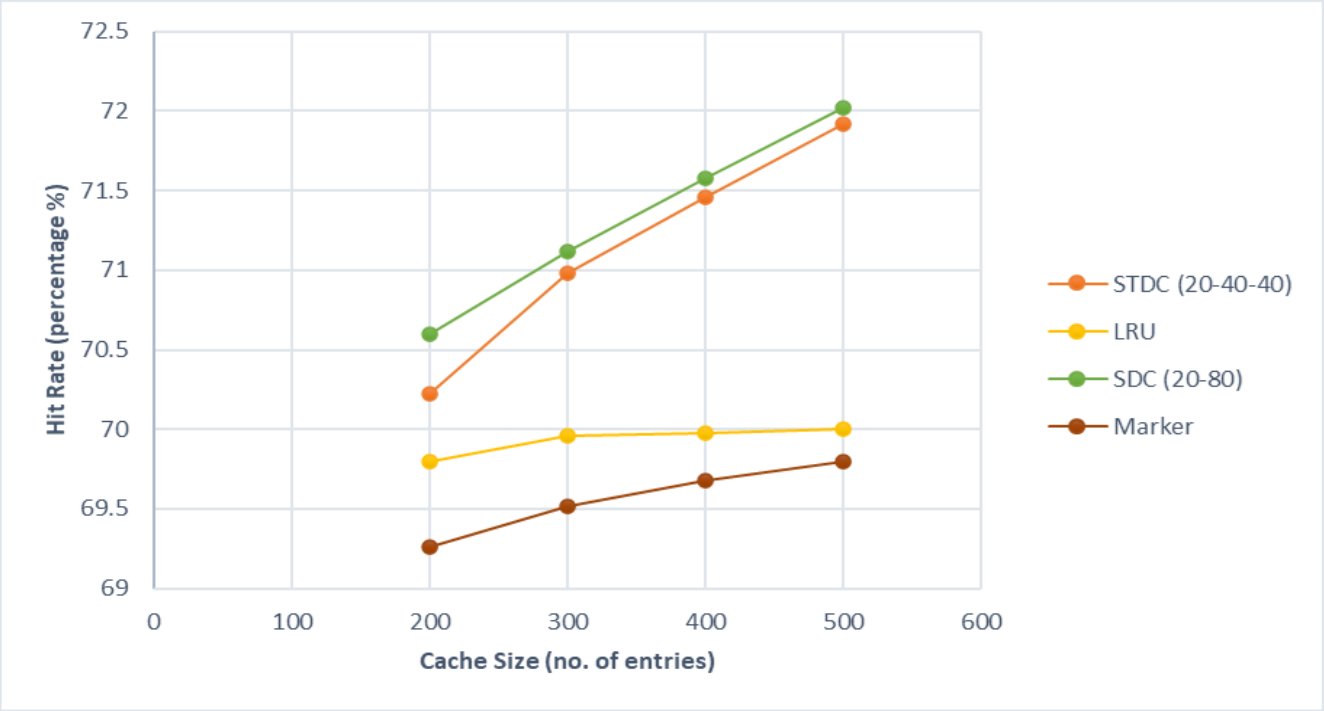
Table 6: Hit rate in percentages (%) for 10K queries and cache size between 200-500 entries

Results

Cache Size	LRU	SDC (20-80)	SDC (30-70)	SDC (40-60)
200	60.19	60.62	60.66	60.75
300	60.37	60.87	61.11	61.25
400	60.52	61.14	61.38	61.6
500	60.97	62.15	62.75	63.36
Cache Size	STDC (20-40-40)	STDC (20-50-30)	STDC (20-60-20)	STDC (20-70-10)
200	60.11	59.8	59.41	59.04
300	60.54	60.28	59.92	59.63
400	60.91	60.73	60.42	60.13
500	61.95	61.86	61.76	61.66

Table 7: Hit rate in percentages (%) for 20K queries and cache size between 200-500 entries

Results



SJSU SAN JOSÉ STATE UNIVERSITY

Figure 12: Hit rate in percentages (%) for 5K queries and cache size between 200-500 entries

Results

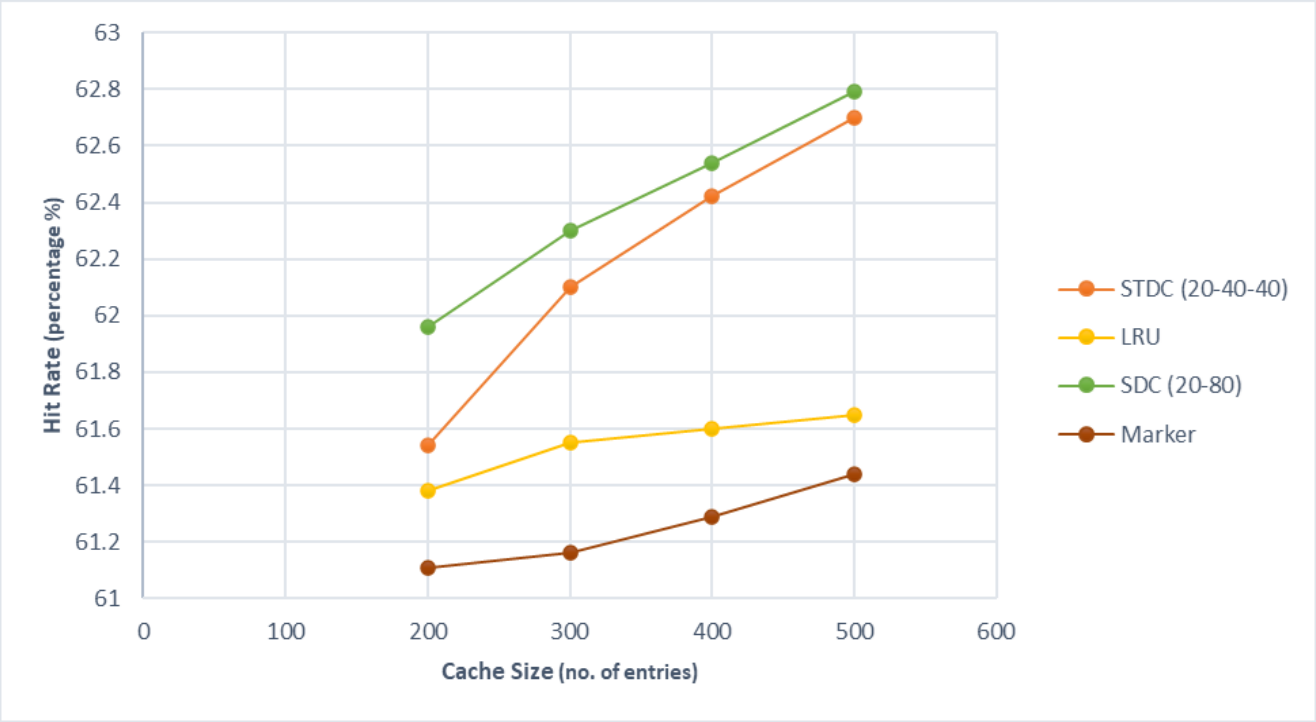


Figure 13: Hit rate in percentages (%) for 10K queries and cache size between 200-500 entries

Results

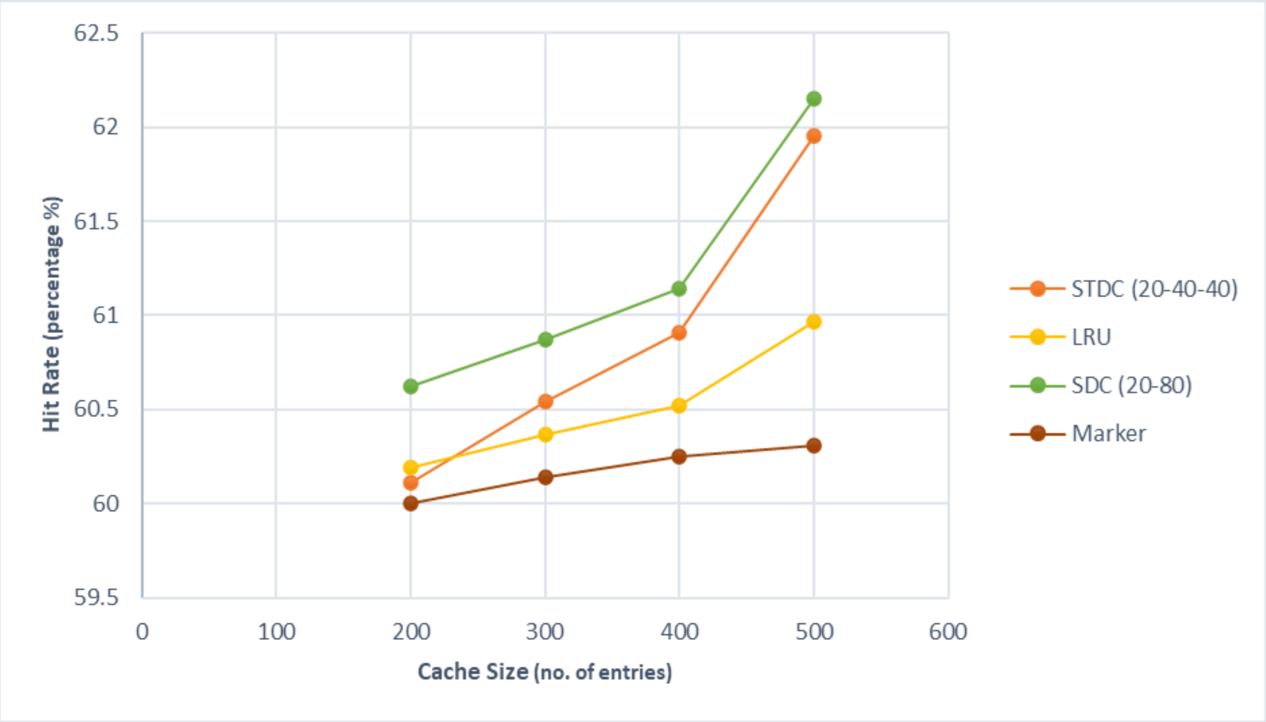


Figure 14: Hit rate in percentages (%) for 20K queries and cache size between 200-500 entries

Conclusion

- Static-Topic-Dynamic Cache performs consistently better than Marker cache and LRU cache
- Performance of Static-Dynamic cache is slightly better than Static-Topic-Dynamic cache
- Static-Dynamic cache shows improvement of 2.3% over Marker cache
- Static-Topic-Dynamic shows improvement of 1.17% over Marker cache
- Users have ability to switch between the implementations based on the requirement

Future Work

- Currently topic model is trained on AOL query logs, it can be changed to use Yioop’s impression data
- Instead of k-means algorithm, other topic models like LDA can be also be implemented
- Topic models also exist which are particularly designed to extract topic from small texts
- Other vectorization method like tf-idf can be used instead of count vectorizer
- Customization feature can also be added to Yioop to customize individual instances of Static-Dynamic and Static-Topic-Dynamic cache

References

[1] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri, “The Impact of Caching on Search Engines,” in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2007, pp. 183–190. doi: 10.1145/1277741.1277775.

[2] E. P. Markatos, “On caching search engine query results,” *Computer Communications*, vol. 24, no. 2, pp. 137–143, 2001, doi: https://doi.org/10.1016/S0140-3664(00)00308-X.

[3] T. Fagni, R. Perego, F. Silvestri, and S. Orlando, “Boosting the performance of Web search engines: Caching and prefetching query results by exploiting historical usage data.,” *ACM Trans. Inf. Syst.*, vol. 24, pp. 51–78, Jan. 2006.

[4] R. Ozcan, I. S. Altingovde, and Ö. Ulusoy, “Static Query Result Caching Revisited,” 2008, pp. 1169–1170. doi: 10.1145/1367497.1367710.

[5] R. Ozcan, I. S. Altingovde, and Ö. Ulusoy, “Cost-Aware Strategies for Query Result Caching in Web Search Engines,” *ACM Trans. Web*, vol. 5, no. 2, May 2011, doi: 10.1145/1961659.1961663.

References

[6] T. Kucukyilmaz, B. B. Cambazoglu, C. Aykanat, and R. Baeza-Yates, “A Machine Learning Approach for Result Caching in Web Search Engines,” *Inf. Process. Manage.*, vol. 53, no. 4, pp. 834–850, Jul. 2017, doi: 10.1016/j.ipm.2017.02.006.

[7] T. Kucukyilmaz, “Exploiting temporal changes in query submission behavior for improving the search engine result cache performance,” *Information Processing & Management*, vol. 58, no. 3, p. 102533, 2021, doi: https://doi.org/10.1016/j.ipm.2021.102533.

[8] I. Mele, N. Tonellotto, O. Frieder, and R. Perego, “Topical result caching in web search engines,” *Information Processing & Management*, vol. 57, no. 3, p. 102193, 2020, doi: https://doi.org/10.1016/j.ipm.2019.102193.

[9] G. Pass, A. Chowdhury, and C. Torgeson, “A picture of search,” Jun. 2006. doi: 10.1145/1146847.1146848.

[10] X. Jin and J. Han, “K-Means Clustering,” in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 563–564. doi: 10.1007/978-0-387-30164-8_425.

Thank you!

Questions?