Big Data Reduction: Lessons Learned From Analyzing One Billion Dota 2 Matches

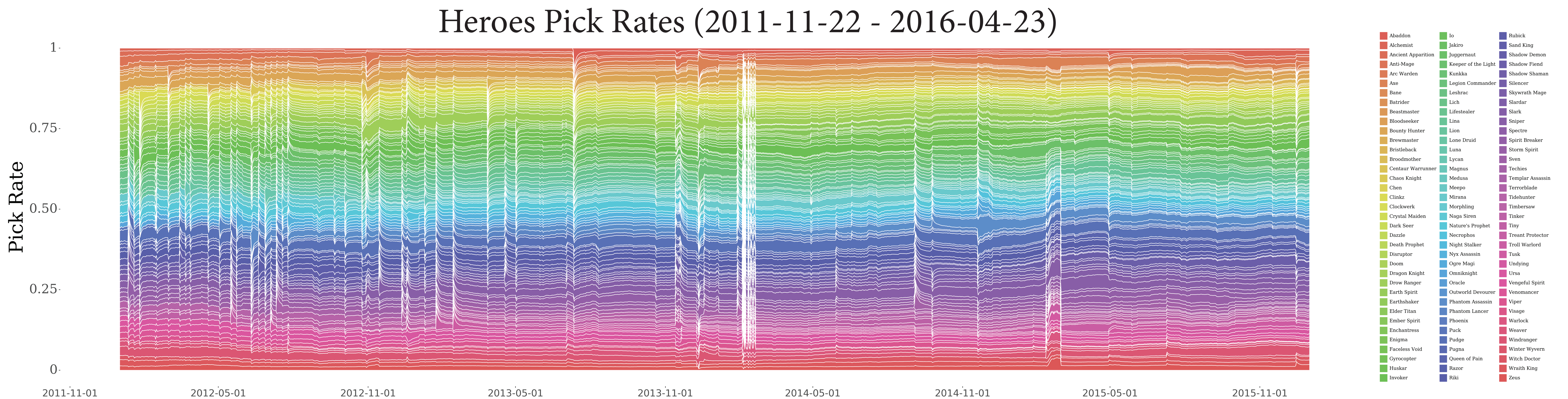
https://git.io/fjJHU

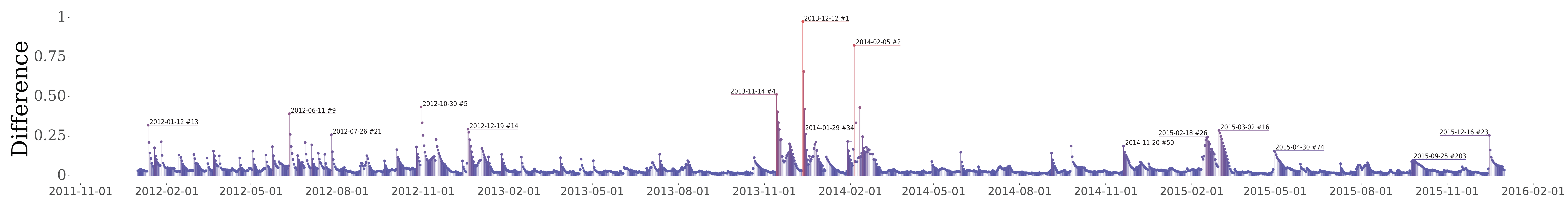
Marco Lussetti
www.marcolussetti.com
marco@marcolussetti.com

Dyson Fraser
www.linkedin.com/in/dysonfraser
dysonfraser@gmail.com

Dr. Mila Kwiatkowska
Supervisor
mkwiatkowska@tru.ca

https://git.io/fjJHU





Why Study Dota 2?

Why study a Video Game? The availability of large datasets of player choices in popular online computer games presents an opportunity to identify sudden changes in choice patterns and explore what factors may contribute to such changes. We were particularly interested in the challenges posed by the scale of such large datasets (we attempted to obtain another >1TB dataset as well).

What is Dota 2? Dota 2 is a popular MOBA (Multiplayer Online Battle Arena) videogame. The core gameplay revolves around teams of five players choosing from a character pool of over 100 characters and facing off against another five-person team in a race to destroy a large structure located in the enemy's base called The Ancient. The game is also known for its large prizepool annual tournament, The International, whose prize pool ranges in the millions of dollars ($25 million for 2018).

Why study Dota 2? Neither of the authors has significant experience with the game, and we had initially explored the availability of datasets for a wide gamut of games with which we have more experience with. However, Dota 2 is unique in having a long-running community project to collect data on the game which is periodically released for use in research. The OpenDota (formerly YASP) project's most recent "data dump" covers >1 billion matches from March 2011 to March 2016 [1] and clearly boosts the appeal of Dota 2 as subject of study.

Research Objectives

Detect metagame shifts from hero pick ratios, that is to attempt to recognize significant changes in players' propensity for picking certain heroes that would be caused by external events (patches to the game, major tournaments, etc.)

"Tame" the dataset's enormity using the simplest tools possible, and using only the resource of a normal machine as may be available to an average researcher without extensive funding

Big Data Reduction

The Dataset. The dataset ranges from March 2011 to April 2016 and contains data on 1,191,768,403 (>1 billion) matches that were played during that time [1]. This data is publicly available as a gunzipped CSV file (151GB zipped, 1.2TB unzipped). At our disposal was only a personal machine with good but not outstanding performance that could not possibly handle the data in its original format. As such, we had to use for Big Data Reduction techniques.

Dimensionality reduction. The curse of dimensionality is a well-known problem where the high number of dimensions present in the dataset cause increasingly high computational burdens [2]. In our case, we have over 50 dimensions in the dataset. We established than only 22 dimensions were needed to achieve our objectives and processed our dataset accordingly. We calculated that on an average day in the dataset, this could have reduced the space needed as much as from 671 MB to 99MB (a 6.7x reduction).

Granularity reduction. We also established that for our purposes of detecting points where the metagame shifted, we need not possess extremely granular data. After all, we are interested in the day or few days when this shift occurs rather than the minute. Thus, we can afford a much coarser granularity than what is present in the dataset – we do not need per-match data, just per-day or per-week summaries of what heroes were picked. A sum of the daily picks (and wins/losses) for each hero was what we sought to produce. Condensing the data in this manner produced extraordinary space savings as expected: the potential reduction could be as great as from 99MB to 3KB (35,545x reduction).

Resulting compression. These are theoretical results for simulated data, however in our actual implementation, we observed the original dataset of 1.2TB being reduced to 3MB or a 396,514x compression. The increased performance in the real-world application is likely a result of JSON compression and some level of data sparsity that was not present in the hand-crafted test case.

Poor Man's Solutions

Java 8 Streams (Bad idea! Keep it simpler!). With such a large amount of data to be processed, it is not viable to load the data in memory, and it is necessary to process it iteratively. Initially, we sought to use Java 8 Streams to do so to avoid having to decompress it ahead of time. However, we quickly discovered that Streams add significant overhead and quickly exhaust a consumer computer's memory. A simple for-loop (well, an iterator really) was much more performant!

CSV Parsers (They matter!). Key to the performance of iterating through such a large dataset. We originally attempted to use the fastest CSV parser available, uniVocity-parsers [3]. However, memory usage from this parser was extremely substantial and would slow down after ~400,000 and crash at ~450,000 records. We fell back to one of the most common and well supported parsers, opencsv, however we found its performance insufficient (around 2,000 lines/s). We found that the second-fastest parser, SimpleFlatMapper, has negligible performance loss but maintains a constant low memory profile. The entire processing could be done with a few hundred megabytes of RAM!

JSON Parsers (Take your pick). As each row contains a JSON field that needs to be parsed, a JSON parser must be employed. We did not see any significant differences in the parsers we briefly auditioned and were able to make our choice by and large based on intuitiveness. We found Jsoniter to be perfectly serviceable.

HashMap Optimization (Doesn't matter). We were concerned about the memory used by the HashMap/Dictionary we intended to store the condensed data in while it was processed, and selected a high-performance library, Trove4J. This library from Palantir stores a THashMap using only (8 * CAPACITY) bytes compared to Java's HashMap that uses (32*SIZE + 4*CAPACITY) bytes [4]. However, it became evident that the HashMaps' memory utilization was not a significant concern because of the coarseness of the granularity.

```
FileReader fileReader = new FileReader(input);
Iterator<String[]> cReader = CsvParser.iterator(fileReader);
while (cReader.hasNext()) parseRow(cReader.next(), onlyCount);
```

Metagame Shifts

The "diversity" graph records the difference between a day's hero picks and the average of the previous 14 days of picks. This is computed using a simple Manhattan distance where each hero's pick rate represents a dimension in the vector.

We sought to match prominent peaks to external events that might influence player behavior such as game patches [5] and tournaments (no effect from tournaments).

2012-01-12 Major rebalancing of heroes & item changes

2012-06-11 Chaos Knight, Phantom Assassin, Gyrocopter released

2012-07-26 Nyx Assassin, Keeper of the Light, Visage released

2012-10-30 Recently released Centaur Warrunner is nerfed

2012-12-19 Major rebalance of most/all champions

2013-11-14 Three Spirits Patch, significant out-of-game & economy changes

2013-12-12 Skeleton King removed shortly before, Legion Commander added, Wraith King added shortly after

2014-01-29 Terrorblade, Phoenix released

2014-02-05 Year Beast Brawl (special game mode)

2014-11-20 Oracle released

2015-02-18 Year Beast Brawl (special game mode)

2015-04-30 Major balance changes

2015-05-03 No major changes, but The Summit 3 Tournament tickets released

2015-09-25 Major balance changes (prev. day)

2015-12-16 Arc Warden released

We believe our approach largely works, even if some major spikes such as that of 2015-05-03 remain unexplained. A SME might be able to help explain such spikes.

Future Work

We have identified several promising venues for extension of the work which we hope to pursue, in addition to overdue cleanup of the codebase which is already in progress.

Exploring further metrics that are available from the underlaying data, chiefly the **wins-losses ratio** for each hero which we currently export but do not explore may be a significant metric which can lead to metagame shift detection.

Another possible avenue for improved detection we have identified is to explore pick ratios not in term of individual heroes but in term of the **role these heroes fill**.

Lastly, the high dimensionality of the dataset means that using as simple a distance function as Manhattan or Euclidean distance is likely not the best idea, and that a **more meaningful dissimilarity measure** might be more effective.

References

[1] The OpenDota Project, "Data Dump (March 2011 to March 2016)," *OpenDota*, 24-Mar-2017. [Online]. Available: https://blog.opendota.com/2017/03/24/datadump2/. [Accessed: 25-Feb-2019].

[2] M. H. ur Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, and S. U. Khan, "Big Data Reduction Methods: A Survey," *Data Sci. Eng.*, vol. 1, no. 4, pp. 265–284, Dec. 2016.

[3] uniVocity Software Pty Ltd, *Comparisons among all Java-based CSV parsers in existence: uniVocity/csv-parsers-comparison*. univocity, 2018.

[4] M. Vorontsov, "Trove library: using primitive collections for performance," *Java Performance Tuning Guide*, 19-Jul-2014. .

[5] Dota 2 Wiki Contributors, "Category:Patches," *Dota 2 Wiki*. [Online]. Available: https://dota2.gamepedia.com/Category:Patches. [Accessed: 25-Mar-2019].