CS:GO Professional Matches and News

Luís Silva*, Mariana Costa* and Pedro Fernandes*
*Faculty of Engineering, University of Porto (FEUP)
up{201503730, 201604414, 201603846}@fe.up.pt

***Abstract*—This paper concerns the collection and analysis of datasets pertaining to the statistics of professional matches in and news related to Counter-Strike: Global Offensive, a Multiplayer First-Person Shooter which has been cimenting its presence in the eSports scene for the past 8 years. HLTV (one of the most recognizable entities in the CS:GO professional scene and responsible for an extensive database of match statistics and news related to the competitions and its actors - players, teams, coaches, etc.) was used as the source for both datasets. The website was scraped in order to collected said datasets, followed by a cleaning and enrichment processes. From here, the data was analysed in order to understand if the data patterns are semantically expected. Finally, the result of this pipeline is a data collection ready to be explored through information retrieval systems, with the aim of answering domain-specific queries.**

***Index Terms*—search systems, data extraction, data refinement, data analysis, Counter-Strike**

I. Introduction

The Counter-Strike series started in 1999 as a Multiplayer First-Person video-game in which two teams - the "Terrorists" and the "Counter-Terrorists" - compete in a series of challenges that involve the "Counter-Terrorists" stopping the "Terrorists" from commiting acts of terror (planting explosives, holding hostages, assassinations, etc.) [1]. Multiple sequels appeared throughout the years, the most recent of which being Counter-Strike: Global Offensive (or CS:GO, as how it will be refered to from this point forward) [2]. As of October 2020, CS:GO has amassed a considerable player base, having an average daily peak of around 870,000 players [3], as well as a very active competitive scene, with prize pools reaching the millions [4] and viewership numbers equally as high [5]. This growth has coincided with the rise in popularity of eSports (of which the broadcast of the CS:GO Major Championship in American television is proof [9]), video game competitions "coordinated by different leagues, ladders and tournaments, and where players customarily belong to teams or other 'sporting' organizations who are sponsored by various business organizations." [6]

Taking into consideration the aforementioned levels of engagement with the competitive leagues of CS:GO, it is to be expected that matches are heavily documented by the community, with varying degrees of attention to detail. Throughout the years, online platforms such as HLTV [13] and Liquipedia [23] have stood out as information hubs that centralize all matters CS:GO-related. Enumerous statistics are collected from the matches, ranging from player kills (e.g. [17]) to heatmaps (visual representations of the location of relevant events that take place during a match - e.g. [18]), and subsequently analysed in a variety of formats (Youtube [19], platforms that aim to help players improve their performance [20], etc.). Given the amount of data this entails, the need for a search system that allows for querying on multiple criteria is substantial (both HLTV and Liquipedia offer said services, with HLTV usually providing the user with more stastistics). The results of these search tasks allow for the previously mentioned analysis to take place, indicating a teams's potential winning trajectory, a listing of the most commonly strategies used in a given map, how players perform when playing as "Counter-Terrorists" or as "Terrorists", among other various topics.

In addition to match statistics, news regarding the game's competitive scene are also frequently produced. While match performances can provide more factual information on a team's current standing, news can highlight other equally relevant (if more subjective) matters, such as a player's controversies [21], player exchanges between teams [22], and much more. As such, they are necessary to obtain the full picture of CS:GO's professional leagues.

The goal of this paper is to describe the preparation of a search system that allows for the user to obtain information on CS:GO matches and news in an expedited manner. The first section presents the necessary steps to obtain and refine the relevant data. The second section gives an overview of the datasets, in order to understand the domain, how the information is distributed, and the retrieval tasks the system will enable.

II. Data Pipeline

In this section, a brief overview on the data sources will be given, as well as a rundown of the main steps behind the cleaning and refinement processes of the two collected datasets.

A. Data Collection

HLTV is an online platform that tracks CS:GO professional matches and offers a way for collaborators to contribute with news pertaining to the video game's competitive scene [13]. It started in 2002 and has since become a hub for Counter-Strike related information. The website counts with nearly 200.000 daily unique viewers [8], and has stood out has one of the most relevant entities in CS:GO journalism. Its importance in the CS:GO scene is recognized by tournaments, who have used their team world rankings for seeding purposes [11], and the game developers themselves, who provide a schedule of professional games in the CS:GO client, using data provided by HLTV [12].

As was previously mentioned, HLTV keeps records of a substantial number of professional CS:GO matches throughout the years, both in minor and major competitions. It is from this database that the dataset on this topic stems from. Since no API is made available by HLTV, the data had to be scraped from the website. The result of this process was a collection of matches held between 2015 and 2020 that was posted on Kaggle (a hub for data scientists and machine learning enthusiasts [7]), and subsequently downloaded in the context of this project [24]. No information on the scraping process was divulged by the author. The dataset was released under a CC BY-NS-SA 4.0 license.

In regards to CS:GO related news, no pre-existing dataset was found; as such, HLTV was scraped in order to obtain all news from 2018 and 2019 using Scrapy, "an open source and collaborative framework" for scraping and web crawling [15]. More information on the scraping process can be found in the annexes. No information regarding scraped content was found on the source (HLTV); however, it is mentioned that copyright to all content on the website is owned by the platform [25].

B. *Data Cleaning and Refinement*

Upon a more thorough analysis of the matches dataset, a number of inconsistencies were detected and handled accordingly. For this cleaning process, OpenRefine was used, a tool for data cleaning and wrangling [14]. The tasks revolved mostly around date formating and removal of columns that were deemed irrevelant to the project at hands. The OpenRefine actions can be consulted in the annexes. In regards to the professional matches, they were additionally filtered to include only entries from 2018 and 2019. As for the news, only the date information was formatted to fit the template adopted when cleaning the matches dataset.

One of the advantages of the match dataset was the inclusion of ID values which allowed the user of said dataset to connect information between the several files it provided (the structure of the data will be detailed in a subsequent section). This opened up the possibility, for example, for different information (from distinct sources) regarding a particular match to be reconciled. However, the news dataset, as it was in its original form, did not provide any way to establish a connection between it and the information on professional matches and its actors. For this to happen, entities needed to be extracted from the news content as a way to establish a bridge between the two domais. From this, a new file which connected news to players and teams would be produced. This posed two problems: which entities should be extracted? And, since there are no limitations as to what a player or team may be called, how will false positives be handled (i.e. a player or team whose name is generic enough to be detected in an abnormal amount of news articles)?

To solve the former, unique player and team names were collected from the dataset on professional CS:GO matches. This list was then fed to spaCy (a NLP tool developed in Python [16]), which proceeded to annotate all ocurrencies of said entities in each news article. The results were then exported to a CSV file.

No systematic solution was found to tackle the problem of false positives. For the data analysis, the entries were reviewed manually; however, that will not be a possibility in the final implemented system.

III. Data Characterization

Having detailed the extraction and refinement processes, the data will now be characterized to a greater extent. An overview of the files that compose the datasets will be provided, including the formats, content structure and number of entries. Then, the conceptual model will be analysed, and finally, some exploratory analysis will be conducted.

The datasets span six CSV files: four pertaining to the match information, and two related to CS:GO-related news. Table 1 describes said files and their structure. Given the extensive nature of some of the files in terms of columns, the specific file structure will be present in the annexes.

A. *Conceptual Model*

The conceptual model can be found in Figure 1. The focus of the project relies on the player, match team, match and match map. The distinction between a team and a match team stems from the desire to avoid a ternary association between team, player and match. A player plays in a match within a specific team; however, throughout a player's career, they might switch teams, and it is important to preserve a player's contractor at the time of a given match. Match teams aim to accomplish just that; they are a "snapshot" of the composition of a team around the time of a given match. A match team is composed of exactly five players and is associated with a single team. A match has exactly two match teams and, consequently, exactly ten players. Said match is played in between one to five match maps (instances of a map within the context of a given match), has a veto process (in which teams choose and exclude which maps will be played) and belongs to a tournament. Each map can be played a variable number of rounds, up to a maximum of thirty. Finally, a news article can mention both players and teams.

B. *Data Analysis*

In order to understand how the data is spread out, and if the data patterns fit the expectations of someone who is familiarized with CS:GO, an analysis of the datasets was in place. The subsequent paragraphs will cover the matches, players and news, respectively.

First, a bar graph representing the number of rounds played in each map during 2018 and 2019 was drawn (Figure 2). A few observations can be made regarding said graph: some maps had significant drops in the number of rounds played between years (i.e. Cache and Cobblestone); this can be attributed to the fact that those maps were removed from the professional map roster sometime in 2019. On the other hand, some maps (i.e. Vertigo) started registering rounds only in

2019 for the opposite reason to the one stated above (this map in particular was introduced to the professional roster only in 2019). As a final remark in regards to the maps, some (such as Mirage and Inferno) are considered ”safer” picks since most teams train on them more frequently; the number of times these maps were picked substantiates that claim.

In regards to the number of matches played on a monthly basis during 2018 and 2019 (Figure 5), one can conclude that periods of relative high activity levels are contrasted with minor slumps in match numbers (e.g July/August of 2018 vs. September/October of that same year). This can be explained by the fact that teams usually have a season break (which is not set to a particular timeframe nor duration; however they might coincide) [10].

A brief look at the news article's character distribution box plot for 2018 and 2019 (Figure 7) indicates that the length of said articles varies little between the two years. The graph was scaled logarithmically since 75% of all news fall under the 2.500 character mark, while the lengthiest articles have around 42.000 and 39.000 characters (for 2018 and 2019, respectively). Outliers are uncommon and usually represent in-depth analysis and overviews of the annual performance of players and teams. Take the ten articles with the highest character count: they are either highlights of top 20 players that year (e.g ”Top 20 players of 2018: dupreeh (5)”) or feature articles on players (e.g. ”From Asia to the world: the story of Bleh”), teams (e.g. ”A year at the summit: how Astralis wrote history”), and the other CS:GO-related topics (e.g. ”Developing in isolation: The story of Australian CS:GO”).

The number of entities and their occurences were also registered in the histogram of Figure 6. During the counting and ordering process, the concerned mentioned in the Data Cleaning and Refinement section was made evident. Figure 6 contains the top 10 entities in news article (during 2018 and 2019). As one can observe, the list is comprised of terms that can easily be misindentified as players or teams (e.g. ”in, ”will”, ”Will”). Since string normalization should not be applied to the extracted teams and players nouns (e.g ”will” and ”Will” may represent different entities), and players with generic names are nonetheless valid, a manual exclusion of entries likely to contain a substantial amount of false positives had to be conducted. Upon the conclusion of this process, the revised top 10 entities is summarized in Figure 7.

C. Possible Search Tasks

Platforms such as HLTV and Liquipedia allow for the search of match statistics, teams, players and maps, as well as news (in the case of HLTV). The proposed search system will provide similar services, with the addition of the possibility for searching for entities within the article text. Table 2 summarizes the aforementioned search tasks.

With these tools at the disposal of the user, some examples of relevant queries include:

- Which player performed better on Inferno in November 2019?
- What is the synopsys of the Astralis vs Liquid match?
- How many times have Astralis and Liquid played each other in 2019?
- Is Astralis a CT-sided team?
- Who is the best support player from the USA?
- How many times have Astralis played Vertigo before the StarLadder 2019
  Major?
- Who's the worst player in Astralis?
- How has Team Liquid fared against better opponents in 2019?
- What team has lowest pick win rate in 2019?
- I want to know more about s1mple.

IV. Conclusions

This paper addressed the collection and characterization of data necessary to implement a search system that can provide the user with a compreensive view of CS:GO Professional Matches (limited to 2018 and 2019). Two datasets were analysed, one for the match statistics and another for the CS:GO-related news. The datasets were cleaned, refined, and subsequently characterized, both conceptually and statistically. Future work includes the implementation of an information retrieval platform that builds upon the collected data.

References

[1] O. Scott. November 27, 2000.”Half-Life: Counter-Strike Review”. [Online]. Available : https://www.gamespot.com/reviews/half-life-counter-strike-review/1900-2657769/. [Accessed: October 25, 2020]

[2] Valve Corporation. 2020. ”Counter-Strike: Global Offensive”. [Online]. Available : https://store.steampowered.com/app/730/CounterStrike_Global_Offensive/. [Accessed: October 25, 2020]

[3] SteamDB. 2020. ”Counter-Strike: Global Offensive”. [Online]. Available : https://steamdb.info/app/730/graphs/. [Accessed: October 25, 2020]

[4] Liquipedia. 4 September, 2020. ”World Electronic Sports Games 2016”. [Online]. Available : https://liquipedia.net/counterstrike/World_Electronic_Sports_Games/2016. [Accessed: October 25, 2020]

[5] Field Level Media. March 2, 2020. ”IEM Katowice sets viewership record amid coronavirus outbreak”. *Reuters*. [Online]. Available : https://www.reuters.com/article/esports-csgo-katowice-viewership-idUSFLM3NTQY6. [Accessed: October 25, 2020]

[6] J. Hamari, M. Sjöblom. 2017. ”What is eSports and why do people watch it?”. *Internet Research*, vol. 27, no. 2. pp. 2.

[7] Kaggle. 2020. ”Kaggle: Your Machine Learning And Data Science Community”. [Online]. Available: https://www.kaggle.com/. [Accessed: October 25, 2020].

[8] Siteworthtraffic. 2020. [Online]. Available: https://www.siteworthtraffic.com/report/hltv.org. [Accessed: October 26, 2020].

[9] ELEAGUE. September 27, 2016. ”ELEAGUE to Host CS:GO Major Championship”. [Online]. Available: https://www.eleague.com/news/2016/9/27/eleague-to-host-csgo-major-championship. [Accessed: October 26, 2020].

[10] LucasAM. August 2, 2019. ”CSPPA announce 2020 summer player break dates”. [Online]. Available: https://www.hltv.org/news/27517/csppa-announce-2020-summer-player-break-dates. [Accessed: October 26, 2020].

[11] MIRAA. October 11, 2020. ”Flashpoint 2 closed qualifier invites revealed”. [Online]. Available: https://www.hltv.org/news/30443/flashpoint-2-closed-qualifier-invites-revealed. [Accessed: October 26, 2020].

[12] Counter-Strike: Global Offensive. [Online]. Available: https://blog.counter-strike.net/index.php/2019/05/24172/. [Accessed: October 26, 2020].

[13] HLTV. 2020. ”CS:GO News & Coverage”. [Online]. Available: https://hltv.org. [Accessed: October 26, 2020].

[14] OpenRefine. 2020. ”OpenRefine”. [Online]. Available: https://openrefine.org/. [Accessed: October 26, 2020].

[15] Scrapy. 2020. ”A Fast and Powerful Scraping and Web Crawling Framework”. [Online]. Available: https://scrapy.org/. [Accessed: October 26, 2020].

[16] spaCy. 2020. ”Industrial-Strength Natural Language Processing”. [Online]. Available: https://spacy.io/. [Accessed: October 26, 2020].

[17] HLTV. 2020. ”Natus Vincere vs. NiP at BLAST Premier Fall Series 2020 — HLTV.org”. [Online]. Available: https://www.hltv.org/matches/2344817/natus-vincere-vs-nip-blast-premier-fall-series-2020. [Accessed: October 26, 2020].

[18] HLTV. 2020. ”BLAST Premier Fall Series 2020”. [Online]. Available: https://www.hltv.org/stats/matches/heatmap/mapstatsid/110809/nip-vs-natus-vincere?showKills=true&showDeaths=false&firstKillsOnly=false&allowEmpty=false&showKillDataset=true&showDeathDataset=false. [Accessed: October 26, 2020].

[19] Hawka. August 2, 2020. ”aRT: The Most Aggressive Player In CS:GO History”. [Online]. Available: https://www.youtube.com/watch?v=s9MbqpTnOh4. [Accessed: October 26, 2020].

[20] Leetify. 2020. ”Leetify - CS:GO Stats & Actionable Insights to help you improve”. [Online]. Available: https://leetify.com/. [Accessed: October 26, 2020].

[21] Professeur. September 1, 2020. ”Heroic suspend hunden following ban”. [Online]. Available: https://www.hltv.org/news/30225/heroic-suspend-hunden-following-ban. [Accessed: October 26, 2020].

[22] LucasAM. October 22, 2020. ”C0NTACT ADD RIGON AND SPINX”. [Online]. Available: https://www.hltv.org/news/30509/c0ntact-add-rigon-and-spinx. [Accessed: October 26, 2020].

[23] Liquipedia. 2020. ”Liquipedia Counter-Strike Wiki”. [Online]. Available: https://liquipedia.net/counterstrike/Main_Page. [Accessed: October 26, 2020].

[24] M. Machado. 2020. ”CS:GO Professional Matches”. [Online]. Available: https://www.kaggle.com/mateusdmachado/csgo-professional-matches. [Accessed: October 26, 2020].

[25] HLTV. 2020. ”HLTV.org Terms”. [Online]. Available: https://www.hltv.org/terms. [Accessed: October 26, 2020].

Appendix A
Glossary

- **Eco round** - in such round, the team chooses to save money by not buying weapons / grenades. Usually this is done with the purpose of having better weapons in the next round;
- **Force buy** - round where the team chooses to spend all of their money despite not having enough money for the theoretically better weapons;
- **Full buy** - the team has enough money to buy the best weapons and grenades;
- **Veto** - CS:GO has 7 active maps. Most professional games have a best of 1, 3, or 5 format, and therefore there needs to be picks and bans, where teams choose which maps they wish, or not, to play;
- **T/CT** - CS:GO has two teams playing against each other on opposite sides. CTs are meant to defend the A and B bomb sites, while Ts want to plant and explode a C4 on one of said bomb sites. After 15 rounds, they will swap. First to 16 rounds wins;

TABLE I
DATASET FILES

Collection	Format	Function
Economy	CSV	Money earned by each team in all rounds of a given map
News	CSV	News collected from HLTV.org
News Entities	CSV	Entities extracted from the scraped news
Picks	CSV	Maps chosen and excluded by the teams in a given match
Players	CSV	Statistics for a given player in a given match
Results	CSV	Statistics for both teams in a given round

TABLE II
SEARCH TASKS

Search for	Restrict on	Order by
Player	Kills Assists Deaths HS Flash Assists KAST KD ADR FKDIFF Rating	Map Team Against Date Interval Side (T/CT) Team Nationality
Teams	Wins Games Played Round Win % Force Buy % Upset Potential Pick Win Rate	Map Team Against Date Interval Side (T/CT)
Matches	Date	Map Teams Date Interval Event
News	Date	Date Interval

Appendix B
Tables

Appendix C
Graphics

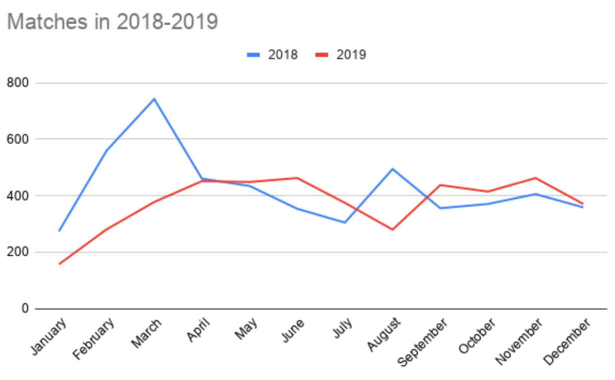


Fig. 1. Matches played in 2018-2019

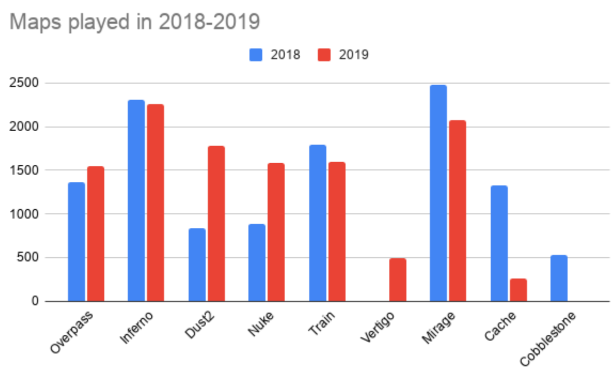


Fig. 2. Maps played in 2018 and 2019

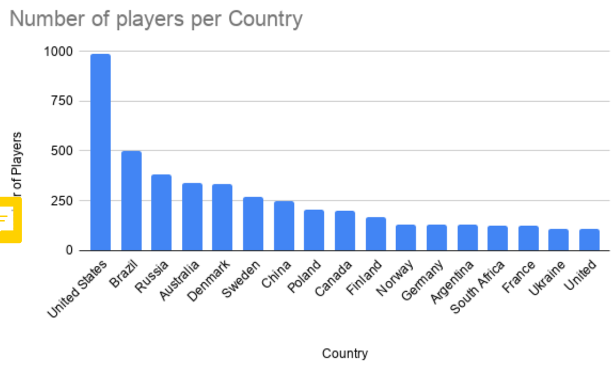


Fig. 3. Number of players per country in 2018-2019

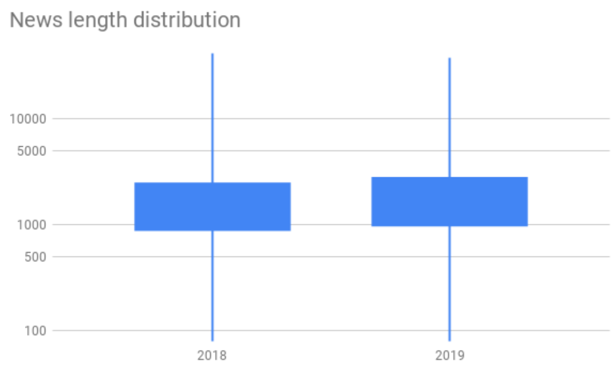


Fig. 4. Number of characters in news in 2018-2019

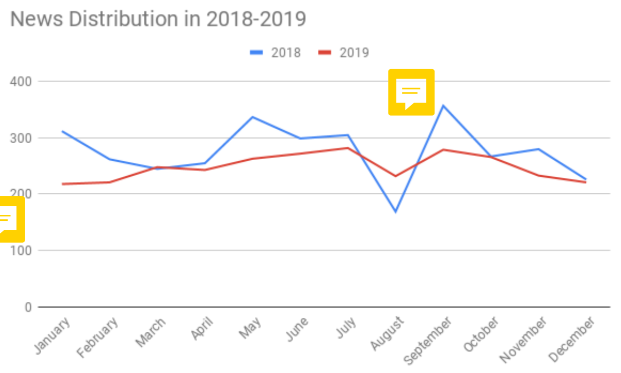


Fig. 5. News distribution in 2018-2019

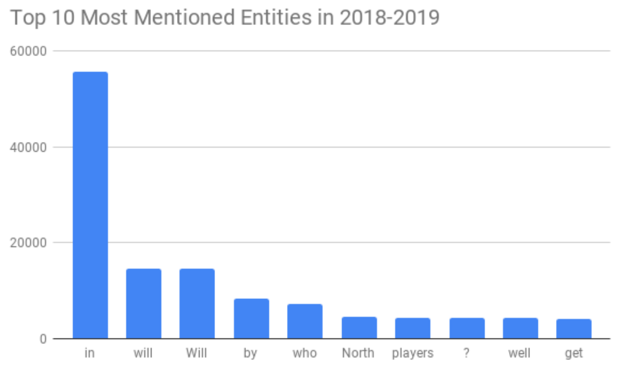


Fig. 6. Top 10 Entities in 2018 and 2019

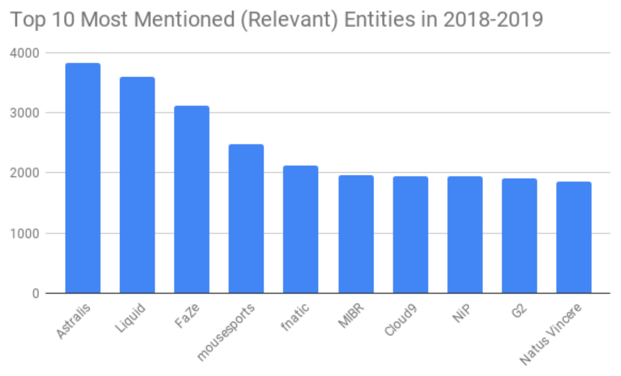


Fig. 7. Top 10 (Relevant) Entities in 2018 and 2019

Appendix D
OpenRefine Actions

```
[
    {
      "op": "core/row-removal",
      "engineConfig": {
        "facets": [
          {
```

```
          "type": "list",
          "name": "_map",
          "expression": "value",
          "columnName": "_map",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": "Default",
                "l": "Default"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "description": "Remove rows"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [],
      "mode": "row-based"
    },
    "columnName": "date",
    "expression": "value.toDate()",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on
 cells in column date using expression
 value.toDate()"
  }
]
```

Listing 1. Economy

```
[
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [],
      "mode": "row-based"
    },
    "columnName": "date",
    "expression": "value.toDate()",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on
 cells in column date using expression
 value.toDate()"
  }
]
```

Listing 2. News

```
[
  {
    "op": "core/column-removal",
    "columnName": "left_over",
    "description": "Remove column
 left_over"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t1_removed_2",
          "expression": "value",
          "columnName": "t1_removed_2",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t1_removed_2",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on
 cells in column t1_removed_2 using
 expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t1_removed_3",
          "expression": "value",
          "columnName": "t1_removed_3",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t1_removed_3",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t1_removed_3 using expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t2_removed_1",
          "expression": "value",
          "columnName": "t2_removed_1",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t2_removed_1",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t2_removed_1 using expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t2_removed_2",
          "expression": "value",
          "columnName": "t2_removed_2",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t2_removed_2",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t2_removed_2 using expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t2_removed_3",
          "expression": "value",
          "columnName": "t2_removed_3",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t2_removed_3",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t2_removed_3 using expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t1_picked_1",
          "expression": "value",
          "columnName": "t1_picked_1",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t1_picked_1",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t1_picked_1 using expression null"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [
        {
          "type": "list",
          "name": "t2_picked_1",
          "expression": "value",
          "columnName": "t2_picked_1",
          "invert": false,
          "omitBlank": false,
          "omitError": false,
          "selection": [
            {
              "v": {
                "v": 0,
                "l": "0.0"
              }
            }
          ],
          "selectBlank": false,
          "selectError": false
        }
      ],
      "mode": "row-based"
    },
    "columnName": "t2_picked_1",
    "expression": "null",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column t2_picked_1 using expression null"
  },
  {
    "op": "core/column-removal",
    "columnName": "inverted_teams",
    "description": "Remove column inverted_teams"
  },
  {
    "op": "core/text-transform",
    "engineConfig": {
      "facets": [],
      "mode": "row-based"
    },
    "columnName": "date",
    "expression": "value.toDate()",
    "onError": "keep-original",
    "repeat": false,
    "repeatCount": 10,
    "description": "Text transform on cells in column date using expression value.toDate()"
  }
]
```

Listing 3. Picks

```
[
    {
      "op": "core/text-transform",
      "engineConfig": {
        "facets": [],
        "mode": "row-based"
      },
      "columnName": "date",
      "expression": "value.toDate()",
      "onError": "keep-original",
      "repeat": false,
      "repeatCount": 10,
      "description": "Text transform on
   cells in column date using expression
   value.toDate()"
    }
  ]
```

Listing 4. Players

```
[
    {
      "op": "core/row-removal",
      "engineConfig": {
        "facets": [
          {
            "type": "list",
            "name": "_map",
            "expression": "value",
            "columnName": "_map",
            "invert": false,
            "omitBlank": false,
            "omitError": false,
            "selection": [
              {
                "v": {
                  "v": "Default",
                  "l": "Default"
                }
              }
            ],
            "selectBlank": false,
            "selectError": false
          }
        ],
        "mode": "row-based"
      },
      "description": "Remove rows"
    },
    {
      "op": "core/text-transform",
      "engineConfig": {
        "facets": [],
        "mode": "row-based"
      },
      "columnName": "date",
      "expression": "value.toDate()",
      "onError": "keep-original",
      "repeat": false,
      "repeatCount": 10,
      "description": "Text transform on
   cells in column date using expression
   value.toDate()"
    }
  ]
```

Listing 5. Results

Appendix E
Dataset File Structure

TABLE III
File Structure

File Name	Columns
Economy	date match_id event_id team_1 team_2 best_of map t1_start t2_start
Players	date player_name team opponent country player_id match_id event_id event_name best_of
Picks	date team_1 team_2 inverted_teams match_id event_id best_of system t1_removed_1 t1_removed_2
Results	date team_2 team_2 map result_1 result_2 map_winner starting_ct ct_1 t_2

Appendix F
Web scraping of CS:GO News

```python
import scrapy

class HLTVNews(scrapy.Spider):
    name = "news"
    count = 0

    def start_requests(self):
        urls =[
            'https://www.hltv.org/news/archive/2018/january',
            'https://www.hltv.org/news/archive/2018/february',
            'https://www.hltv.org/news/archive/2018/march',
            'https://www.hltv.org/news/archive/2018/april',
            'https://www.hltv.org/news/archive/2018/may',
            'https://www.hltv.org/news/archive/2018/june',
            'https://www.hltv.org/news/archive/2018/july',
            'https://www.hltv.org/news/archive/2018/august',
            'https://www.hltv.org/news/archive/2018/september',
            'https://www.hltv.org/news/archive/2018/october',
            'https://www.hltv.org/news/archive/2018/november',
            'https://www.hltv.org/news/archive/2018/december',
            'https://www.hltv.org/news/archive/2019/january',
            'https://www.hltv.org/news/archive/2019/february',
            'https://www.hltv.org/news/archive/2019/march',
            'https://www.hltv.org/news/archive/2019/april',
            'https://www.hltv.org/news/archive/2019/may',
            'https://www.hltv.org/news/archive/2019/june',
            'https://www.hltv.org/news/archive/2019/july',
            'https://www.hltv.org/news/archive/2019/august',
            'https://www.hltv.org/news/archive/2019/september',
            'https://www.hltv.org/news/archive/2019/october',
            'https://www.hltv.org/news/archive/2019/november',
            'https://www.hltv.org/news/archive/2019/december',

        ]
        for url in urls:
            yield scrapy.Request(url=url, callback=self.parse_search_page)

    def parse_search_page(self, response):
        for link in response.css('a.article'):
            next_article = link.css('a::attr(href)').get()
            next_article = response.urljoin(next_article)
            yield scrapy.Request(next_article, callback=self.parse_article)


    def parse_article(self, response):
        self.count += 1
        text = response.css('div.newsdsl .legacy-con p > span::text, div.newsdsl .legacy-con p > span > a::text').getall()
        collapsed_text = ' '.join(''.join(text).split())
        yield {
            'ID': self.count - 1,
            'date': response.css('div.article-info .date::text').get(),
            'title': response.css('h1.headline::text').get(),
            'text': collapsed_text,
            'author': response.css('div.article-info .author a span::text').get(),
        }
```

Listing 6. News

Appendix G
Entity Extraction

```python
import csv

players = set()
teams = set()
maps = set()

with open('../../data/hltv/players.csv', encoding='utf-8') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            players.add(row[1])
            teams.add(row[2])
            line_count += 1
    print(f'Processed {line_count} lines.')
    players_unique = list( dict.fromkeys(players) )
    teams_unique = list( dict.fromkeys(teams) )
    print(f'Number of players: {len(players_unique)}.')
    print(f'Number of teams: {len(teams_unique)}.')

with open('../../data/hltv/results.csv', encoding='utf-8') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            maps.add(row[3])
            line_count += 1
    print(f'Processed {line_count} lines.')
    maps_unique = list( dict.fromkeys(maps) )
    print(f'Number of maps: {len(maps_unique)}.')
    print(f'Maps: {", ".join(maps_unique)}')

with open('../../data/results/players.csv', mode='w', newline="", encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=',', quotechar='"', quoting=csv.QUOTE_MINIMAL)
    for player in players_unique:
        if player:
            csv_writer.writerow([player])

with open('../../data/results/teams.csv', mode='w', newline="", encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=',',
     quotechar='"', quoting=csv.QUOTE_MINIMAL)
    for team in teams_unique:
        if team:
            csv_writer.writerow([team])

with open('../../data/results/maps.csv', mode='w',
    newline="", encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=',',
     quotechar='"', quoting=csv.QUOTE_MINIMAL)
    for map in maps_unique:
        if map:
            csv_writer.writerow([map])
```

Listing 7. Entity Extraction

```
import spacy
from spacy.matcher import PhraseMatcher
from spacy.tokens import Span
import csv

players = set()
teams = set()
maps = set()

news = set()

tags = set()

with open('../../data/results/players.csv', encoding
    ='utf-8') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        players.add(row[0])

with open('../../data/results/teams.csv', encoding='
    utf-8') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        teams.add(row[0])

with open('../../data/results/maps.csv', encoding='
    utf-8') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        maps.add(row[0])

print(f'Number of players: {len(players)}.')
print(f'Number of teams: {len(teams)}.')
print(f'Number of maps: {len(maps)}.')

nlp = spacy.load('en_core_web_sm')

playerMatcher = PhraseMatcher(nlp.vocab)
playerPattern = [nlp(player) for player in players]
playerMatcher.add('PLAYER', None, *playerPattern)

print(f'Processed PLAYER entities.')

teamMatcher = PhraseMatcher(nlp.vocab)
teamPattern = [nlp(team) for team in teams]
teamMatcher.add('TEAM', None, *teamPattern)

print(f'Processed TEAM entities.')

mapMatcher = PhraseMatcher(nlp.vocab)
mapPattern = [nlp(map) for map in maps]
mapMatcher.add('MAP', None, *mapPattern)

print(f'Processed MAP entities.')

with open('../../data/news.csv', encoding='utf-8')
    as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        news.add(tuple([row[0], row[3]]))

print(f'Number of news: {len(news)}.')

for article in news:
    doc = nlp(article[1])
    matches = playerMatcher(doc)
    doc.ents = []
    for match_id, start, end in matches:
        span = Span(doc, start, end, label=match_id)
        tags.add(tuple([article[0], span, 'PLAYER'])
    )
    matches = teamMatcher(doc)
    doc.ents = []
    for match_id, start, end in matches:
        span = Span(doc, start, end, label=match_id)
        tags.add(tuple([article[0], span, 'TEAM']))
    matches = mapMatcher(doc)
    doc.ents = []
    for match_id, start, end in matches:
        span = Span(doc, start, end, label=match_id)
        tags.add(tuple([article[0], span, 'MAP']))
    print(f'Processed article {article[0]}.')

with open('../../data/results/newsTag.csv', mode='w'
    , newline="", encoding='utf-8') as csv_file:
    csv_writer = csv.writer(csv_file, delimiter=',',
     quotechar='"', quoting=csv.QUOTE_MINIMAL)
    for tag in tags:
        if tag:
            csv_writer.writerow(tag)

print(f'newsTag.csv created.')
```

Listing 8. Tagging

Appendix H
Description of Counter-Strike: Global Offensive Mechanics

CS:GO is the most popular shooting game in the market, mainly because of its simple yet hooking mechanics. The player starts the game as part of a team of 5, and is either a Terrorist or Counter-Terrorist. In each round, the terrorists must try to plant and explode a bomb in one of two bombsites or kill the other team, while the counter-terrorists have to stop their adversaries, by killing them or defusing the bomb. This goes on for 15 rounds in the first half, then the teams switch sides: first to 16 rounds win.

In order to kill opponents, players need to buy weapons at the beginning of the round. They must choose carefully what to buy since their money is limited, and as expected the better weapons are more expensive. Apart from this, players can also purchase equipment such as armour, defuse kits and grenades (smoke, flashbang, high explosive or incendiary), in order to gain situational advantages.

The economy is one of the main problems in CS:GO. If a team doesn't know how to manage it, they'll most likely lose the match. In the first round of the game, each player starts with a pistol and 800 dollars. This is enough money to buy light armour, grenades or an upgraded pistol. When a team wins a round, they'll receive around 3000 dollars. When they lose, they still receive a bonus, that increases with each consecutive round that is lost. However, when a losing streak is broken, the bonus for losing future rounds decreases as well. Therefore, depending on the situation, teams may have to make tough decisions. If they're winning a lot of rounds their economy should be great (unless they can't survive the rounds with more than 2 players alive, in that case they have to keep rebuying equipment). When they lose a round, they have to see how much money they have, and decide whether they should buy or not. Additionally, they may also need to make mid-round decisions: if a player is, for example, in a 1 versus 4 situation, he'll most likely decide to save his equipment into the next round, while the opposing team may try to hunt him so that he can't keep anything.

Finally, each game of Counter-Strike takes place in a map: a contained world, that normally contains two locations on opposite sides where the teams spawn in each round, and two bombsites where the terrorists try to plant a bomb. Once again, the concept is simple, yet most of the maps have their own identity, something that separates them from others, be it their layout, geographical/historical context, sound queues or even colors.

Appendix I
Data Pipeline

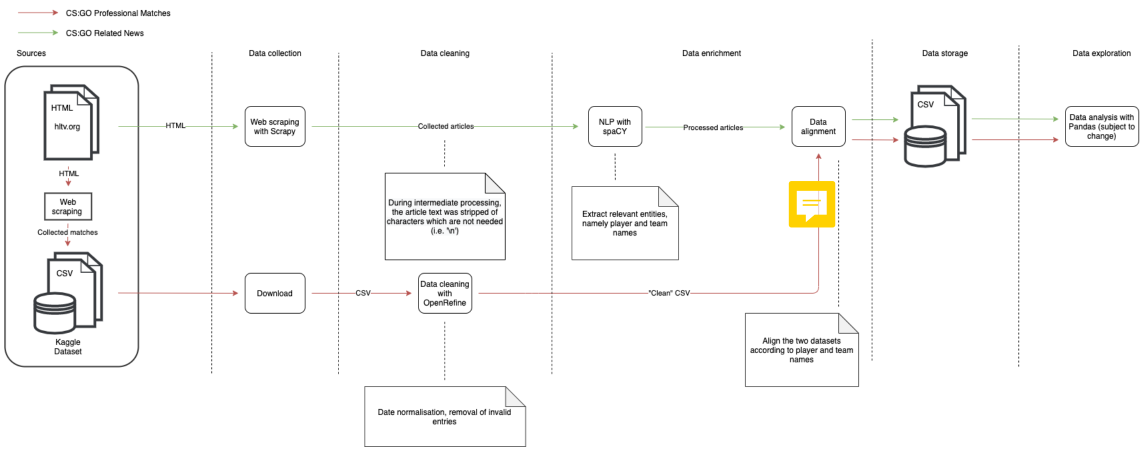
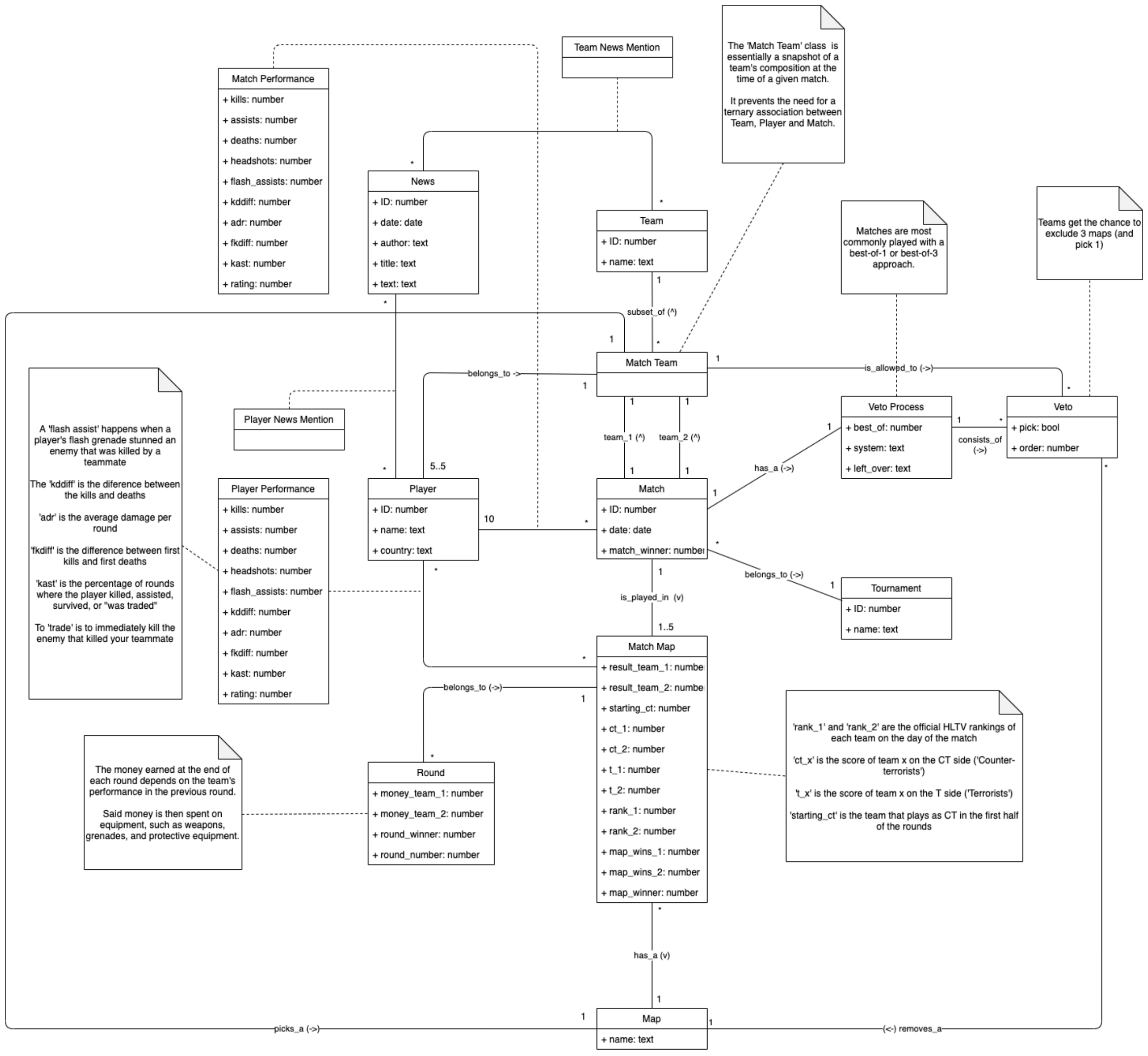

CS:GO Professional Matches
CS:GO Related News
Sources
Data collection
Data cleaning
Data enrichment
Data storage
Data exploration
HTML
hltv.org
HTML
Web scraping
Collected matches
CSV
Kaggle Dataset
HTML
Web scraping with Scrapy
Collected articles
NLP with spaCY
Processed articles
Data alignment
CSV
Data analysis with Pandas (subject to change)
During intermediate processing, the article text was stripped of characters which are not needed (i.e. '\n')
Extract relevant entities, namely player and team names
Download
CSV
Data cleaning with OpenRefine
"Clean" CSV
Align the two datasets according to player and team names
Date normalisation, removal of invalid entries

Fig. 8. Data pipeline

Appendix J
Conceptual Model



The 'Match Team' class is essentially a snapshot of a team's composition at the time of a given match.

It prevents the need for a ternary association between Team, Player and Match.

Matches are most commonly played with a best-of-1 or best-of-3 approach.

Teams get the chance to exclude 3 maps (and pick 1)

A 'flash assist' happens when a player's flash grenade stunned an enemy that was killed by a teammate

The 'kddiff' is the diference between the kills and deaths

'adr' is the average damage per round

'fkdiff' is the difference between first kills and first deaths

'kast' is the percentage of rounds where the player killed, assisted, survived, or "was traded"

To 'trade' is to immediately kill the enemy that killed your teammate

The money earned at the end of each round depends on the team's performance in the previous round.

Said money is then spent on equipment, such as weapons, grenades, and protective equipment.

'rank_1' and 'rank_2' are the official HLTV rankings of each team on the day of the match

'ct_x' is the score of team x on the CT side ('Counter-terrorists')

't_x' is the score of team x on the T side ('Terrorists')

'starting_ct' is the team that plays as CT in the first half of the rounds

Fig. 9. Conceptual model