Ontology-based Knowledge System and Team Verification Tool for Competitive Pokémon

Daniel Verdear and Ubbo Visser
University of Miami, Department of Computer Science
1365 Memorial Drive
Coral Gables, FL, 33146
{verdear|visser}@cs.miami.edu

Abstract

Competitive Pokémon is a domain with rich semantics and complex relationships between its elements. Current research in the domain has focused on developing AI agents to select moves within a match, ignoring the problem of team building. We propose a team verification tool based on ontologies to model the complexities of the domain. A user can input their team into the tool, and the tool uses a description logic reasoner to classify Pokémon into their appropriate roles. The tool exports a visual representation of the team and a score evaluating its competitive viability. The classifications made by the TeamVerify tool have 87.7% precision and 86.0% recall in a multiclass, multilabel domain. We expect such a tool to reduce the learning curve for novice players who have not yet built intuitions on proper team structure.

Introduction

Nintendo's Pokémon is the third best-selling video game franchise of all time, with 24 main series releases since 1996 (The Pokémon Company 2020). As the games' popularity grew, communities of players began organizing tournaments for the game's player-vs-player battle mode. This competitive community flourished because of the game's strategic elements and its frequent updates keeping the game from becoming stale or boring. Competitive Pokémon battling shares many structural similarities to chess, a game that is the subject of influential AI research. Both are turn-based games where individual pieces are constrained in their potential moves. Similarly, tactical decisions are made by synthesizing hard knowledge about a player's own strategy and soft knowledge inferred from observing their opponent.

However, unlike in chess, Pokémon players are allowed to select their six Pokémon from an ever-growing roster (currently set at 893) and select each Pokémon's four moves from a list of dozens of potential moves. Although competitive Pokémon is a two-player game with each player having up to nine potential moves (four potential attacks and five potential switches), there are numerous player-chosen variables that can greatly affect the outcome of each move and its state transition. This complicates the procedure for calculating the state space of a competitive Pokémon match. Even by making perfect plays on every turn, a player or AI agent may still lose because their team is outclassed by their opponent's. Another consideration for AI systems playing Pokémon is that a significant amount of information is leaked before the first turn. In a chess match, an opponent's strategy must be determined by examining their moves. The six Pokémon on a player's team can reveal many details on that player's planned strategy, and failing to understand this information will consistently disadvantage AI agents.

In most contexts, a decision-making system based on information storage and retrieval would be built on top of a database system. Building a database system for this domain would require hard-coding many key concepts, which decreases their generality. We hypothesize that using a knowledge-based system will lead to a more robust tool capable of inferring key concepts and allowing the system to generalize better than a database system.

Related Works

Competitive Pokémon asks two fundamental research questions, the first of which is whether you can decide a player's next move given each player's team and the sequence of moves leading up to the current turn. Current academic research is primarily focused on this question (Simões et al. 2020; Huang and Lee 2019). Researchers benefit from the discrete, turn-based context of Pokémon battles when designing their AI systems. However, this project tackles the second research question posed by competitive Pokémon.

This larger problem of building an effective and versatile team of six Pokémon has been ignored through assumption. Previous works on tactical AI agents have assumed that the team being used is optimal, but for nearly all but the highest-level tournament games, this property should not be assumed (Huang and Lee 2019). Additional work on automated team building has been done outside of the academic realm, with several tools being developed and shared in online forums (DigitalFlow 2019; Pyrotoz 2020; Sciarrone 2020). These tools are built on simple heuristics rather than semantic understanding of the game.

Before even playing a single match, a player must decide which six Pokémon to use on their team. This poses a challenge in knowledge modeling and storage, as the universe of all competitively viable Pokémon has complex relationships that are essential to model in order to build an effective team. These relationships are fluid; for example, using

the same Pokémon with a different item may change its relationships to its teammates. Ontologies have been used extensively to store knowledge in a format that encourages complex relationships between entities (Niaraki and Kim 2009; Toledo et al. 2011). These techniques have not been introduced to any esports context, but the complexity of most competitive video games lends itself to the strength of ontological modeling.

General Approach

We propose to develop an ontology-based team verification tool (TeamVerify). The tool takes as input a competitive Pokémon team expressed in the official Smogon team string format. The team string format provides all information necessary to uniquely identify an instance of a Pokémon, and an example of the representation is available on github. For each Pokémon on the user's team, the tool compares the given information to the classes defined within the ontology and places each Pokémon in the roles that they fill. It then outputs relevant information to the player, such as the role membership lists, and allows the player to have a quantitative measure of their team's viability within seconds. The knowledge base is written as an ontology in OWL2-DL, and the reasoning system used for inferences is HermiT, which has been shown to optimize the tableau calculus required for description logic (Shearer, Motik, and Horrocks 2008; Glimm et al. 2014).

The Pokémon domain is comprised of numerous, often overlapping, classes of Pokémon. In most cases, the class membership of a Pokémon can be determined using properties lifted from a Pokémon's set (the unique combination of a species, four moves, an ability, an item, and a distribution of the Effort Value stats). These initial classifications can then be used to infer a more complex classification. The TeamVerify tool reasons on input rather than access a database of pre-inferred information, so description logic is chosen over first-order logic. Computational efficiency is a primary goal of TeamVerify, and there should be no contexts where a reasoning task keeps the tool from terminating. Despite this focus on complexity, description logic is still preferred over less complex logics because of Pokémon's use of quantifiers and other logical concepts. Less complex logics are unable to model the domain fully (Baader et al. 2003). Because of these characteristics of the domain, description logic was chosen as the method for team annotation. Similarly, modeling the domain as an ontology is a natural knowledge representation. Ontologies are also well-suited for organizing knowledge that may change over time. As will be discussed with the data domain, the concern of shifting knowledge is well-founded.

The current state of competitive team generation is not rigorous. Applications rely on heuristics to make decisions about team structure, and the popularity of certain applications changes frequently as the data for a previous tool becomes outdated. Therefore, it is an open question whether automated team generation for Pokémon is even a solvable problem. Using the TeamVerify tool, we seek to discover whether the problem is verifiable, whether software can decide which of two teams is more competitively viable. TeamVerify stops short of being a full generator, while still providing key features that are useful to players.

The Pokémon Ontology

As mentioned earlier, a Pokémon set includes the unique combination of a Pokémon species, four moves, an item, and an ability. Each of these elements is required for inference. These four disjoint classes form the base of the ontology. Items and Abilities need no further segmentation, and Moves require a simple segmentation between Stat-Boosting moves and all others. The hierarchy of Pokémon involves the most segmentation, and most of these segmentations require inferences.

Any instance of a Pokémon is classified into the general Pokémon class. This level of the ontology checks that each instance has the four elements of a valid set. A Pokémon is then put into two independent classes, one corresponding to its typing and another, Offense or Defense, corresponding to its Effort Values. The Offense and Defense classes are then divided further into classes for each of the Primary Roles, or what the Pokémon contributes to a strategy. The Secondary Roles, additional useful traits, involve integrating information from each of these previous classifications, and each secondary role is a direct descendant of the Pokémon class. With this taxonomy in place, asserting a Pokémon based on its team string data classifies it with the classes relevant for TeamVerify. The result of this modeling step is an ontology with 61 classes, 14 object properties, 7 data properties, and an expressivity of ALCROIQ(D), showing the degree to which the domain is connected (github link to ontology).

The strategy of data collection and data assumptions have the final goal of populating a Pokémon ontology that can be used to model the complexities of the competitive Pokémon domain. A robust ontology allows for a reasoning system to recognize patterns in user-provided team data. The Pokémon ontology used in this project was manually developed using the domain knowledge of the first author and from a survey of high-level Pokémon tournament players. Every player was asked how they defined each of the essential roles and what they viewed as the defining characteristics of each role. Their responses were aggregated and used to develop the class definitions in the ontology.

Pre-Loaded Knowledge

Aside from the class hierarchy, the ontology requires a certain amount of pre-loaded knowledge to facilitate proper classifications. As will be mentioned in greater detail in the following section, the roster of top-tier competitive Pokémon is recalculated monthly. Each month, the ontology is refreshed with the new list of Pokémon, and this affects the determination of what common Pokémon should be addressed by a team. This constant refreshing allows for the re-definition of certain secondary roles, mainly counters to common offensive and defensive Pokémon. This feature calls upon previous research in the fields of ontology versioning and provenance. The versioning protocol has not yet been implemented, but we intend to follow the insights of (Plessers and De Troyer 2005) which propose versioning

protocols. Some information is also unique to each species of Pokémon but not included in the team string. Data such as typing or speed stats are pulled from online resources and asserted to the ontology.

Most of the raw data obtained in the data collection phase is stored as individuals in the ontology. All Pokémon species, items, moves, abilities, and typings are pre-loaded into the ontology. Using individuals to model scraped data also allows set notation to be used in class definitions. For example, the Sweeper role is dependent on having a move that boosts a Pokémon's speed. Since moves are stored as individuals, the set of speed-boosting moves can be enumerated, leading to a more efficient ontology on which to run the HermiT reasoner.

The relationships between individuals are modeled as object properties. Each Pokémon can hold a single item, so a Pokémon can be linked to at most one Item with the hasItem object property. Similar properties exist for linking a Pokémon to its four moves, a Pokémon to its Ability, or a Pokémon to the types it is weak to. This is not an exhaustive list of all object properties but provides a high-level overview of how object properties are used to model complex relationships within this domain.

The result of this modeling step is a knowledge base of Pokémon, each semantically linked to the set of their moves, item, ability, base speed stat, and EV distribution. The size of this base ontology is 4,430 axioms that connect 214 individuals with 14 object properties and 7 data properties. As noted previously, there are 61 classes into which an individual can be assigned. However, a majority (3,141) of the axioms are object property assertions asserted or inferred from the pre-loaded knowledge.

Data Domain and Collection

The goal of the data collection phase is to populate the Pokémon ontology with information that is not available in the Smogon team string format but is necessary for determining what roles a Pokémon fills. A Pokémon record includes: Species name, Typing, Base Statistics (Hit Points, Attack, Defense, Special Attack, Special Defense, Speed), and the Smogon tier it is a member of. The content of these records is unique to each species of Pokémon and is not subject to user-chosen variability. Therefore, for each Pokémon, the record information can be gathered and used to pre-populate the ontology without concern for potential changes. This information is collected automatically from Smogon web resources using a scraper script written in Python (Smogon community 2020b).

Smogon Usage-based Tiering

The Smogon community developed a usage-based tiering approach to classify Pokémon based on their competitive viability (Smogon community 2020a). This approach functions under the assumption that players will generally use the best Pokémon available, and this asumption allows usage rates to act as a proxy for viability. Except in cases where newly released Pokémon need to be tiered quickly, usage rate is calculated using a weighted average of usage over the preceding three month period.

Tiers are defined using a simple heuristic, which can be translated to inferential statistics, that a Pokémon is classified into a tier if there is a 50% chance it will appear in at least one of 15 randomly selected games. Tiering begins with the OverUsed (OU) tier, where all Pokémon are allowed. Once sufficient data is available, a tier cutoff is established - typically 4.52% usage - and all Pokémon above that threshold are classified as OU (Smogon community 2020c). Remaining Pokémon are added to the subsequent UnderUsed (UU) tier, and the process is repeated until all Pokémon are tiered.

The scope of this project is teams built for the OverUsed tier. Therefore, our ontology is only preloaded with data from Pokémon in the OverUsed and UnderUsed tiers. Players in the OverUsed tier are allowed to use Pokémon from the OverUsed tier or any lower tiers. However, we make the assumption that Pokémon in tiers lower than UnderUsed are outclassed by higher tiered Pokémon in nearly all cases.

Data Assumptions

Applying description logic to an ontology knowledge system is an exponential time algorithm (Baader et al. 2003). Therefore, since the result of this project is a user-facing application, several assumptions about the data are made in order to limit the practical running time of the reasoner. The primary assumption made is described in the preceding subsection, where omitting tiers below UnderUsed reduces the number of Pokémon from 890 to approximately 100.

The tool assumes that the user will be inputting a valid team. If the user inputs a team with fewer than six Pokémon, the tool issues a warning and continues without addressing this specific issue. If the user inputs a Pokémon without an item or with fewer than four moves, the tool terminates while parsing. We also assume that the user knows a Pokémon's optimal candidate roles. For example, if a Pokémon has very high defenses and support capabilities, but the user chooses to equip them with offensive moves, an offensive item, and and offensive ability, the tool is not able to recognize that mistake and instead treats the Pokémon as an offensive Pokémon.

The TeamVerify Tool

The result of this project is a Python package, available on github, that includes the command-line team verification tool as well as several scripts used to populate the ontology upon which the tool is based. The tool uses the Owlready2 Python library (Lamy 2017) to interact with the ontology, allowing users to interface with a simple command-line program rather than an ontology manager such as Protégé (Noy et al. 2001). Aside from the trivial task of transforming string input into a Python dict, there are two primary features that TeamVerify provides. Each will be discussed briefly.

Interface with the Ontology

Once the user-inputted data is processed, each of the six Pokémon is asserted into the ontology as an individual. If the team has any syntactic flaws, the tool alerts the user and halts

Table 1: Results of TeamVerify: automatic classification compared to manually labeled data

	Classification	Primary Roles	Secondary Roles	Average Per Team	Total
Ground-Truth	186	267	616	34.5	1,069
Predicted	184	242	622	33.8	1,048
Ratio	98.9%	90.6%	100.9%	97.9%	98.0%

gracefully. Given a syntactically sound set of data, the ontology reasoner is then called. TeamVerify uses the optimized HermiT reasoner with a memory allocation of 2GB, and the process of completing all annotations takes 40.4 seconds on average (std=9.4) using an Intel i5 processor with 1.80GHz clock speed. Following the rules outlined in the description of the class hierarchy, each role is filled during the reasoning process, and this populates membership lists for each primary and secondary role. This set of membership lists is then passed to the functions that generate output. An example of one type of inference can be seen in Figure 1. After adding a new individual and asserting *hasSpecies specChandelure* under object properties, the system infers that the individual *isResistantTo tyFire*.

Scoring and Generating Output

TeamVerify provides output through the command-line and in a text file report. The report includes a copy of the team string submitted by the user, membership lists for each of the essential roles, and a team score that quantifies the team's viability. At the end of the report, TeamVerify displays a list of suggestions it has reasoned from the knowledge base. For any essential role that is empty, TeamVerify will query the knowledge base to find a suitable candidate. Additionally, TeamVerify identifies the Pokémon that fills the fewest roles and suggests that it be replaced.

The membership lists are generated using the DL Query feature of OWL2. A DL query allows for a defined class that contains reasoned information to be queried for basic inferences such as member instances. The list of Pokémon filling each role follows trivially from this feature. Team suggestions are executed using SPARQL queries. The more formal structure of SPARQL queries allow complex querying based on membership in an inferred class and filtered by number of occurrences.

The defining feature of TeamVerify, that sets it apart from a simple ontology interface, is team scoring. Once a team's membership lists have been inferred, these lists are used to calculate a score that roughly translates to a team's competitive viability. For each role in TeamVerify, there is a membership list of length n and the score for that role is $\sum_{i=1}^{n} 1/i$. The purpose of this summation is to provide diminishing returns for redundancies in team building. If two Pokémon are used exclusively for a single role, their score will be reduced to an average of 0.75. This scoring forces redundant Pokémon to provide versatility or have their impact greatly reduced. An exception to this scheme is primary roles such as Wallbreaker or Sweeper, where having multiple win conditions provides versatility in overall strategy. For a membership list of length n for these roles, the score is n. The sum of these role scores is the overall team score.

Evaluation

We have evaluated the TeamVerify tool from two perspectives, whether its classifications (and therefore its team scores) are accurate and whether its scores are indicative of more competitively viable teams.

Evaluating Classifications

A set of 31 competitive Pokémon teams were gathered from Smogon tournament forums (such as the private The Academy forums on Discord) (The Academy community 2020). The data was manually annotated by a team of two experienced players, and the final dataset contains 1,069 annotations. Within these annotations, 186 contain Pokémon classifications, 267 primary roles, and 616 secondary roles. The raw team strings for these 31 teams were inputted into the TeamVerify tool, and a summary of the results is shown in Table 1.

We can see that misclassifications are present in each major category that the TeamVerify tool tests. The automated classification correctly labels 920 examples, for a precision of 0.8778 and recall of 0.8606 (F1 score = 0.8691). Several quirks in the TeamVerify reasoning system are the cause of these misclassifications. For primary roles, the decision boundary separating Defensive Walls from Defensive Pivots requires further refinement as several Walls were erroneously classified as Pivots. Additionally, the union of all Wallbreakers, Sweepers, and Offensive Pivots does not perfectly equal the set of possible Offensive Pokémon. This leads to an Offensive Pokémon running a niche strategy not being classified within a primary role.

Some secondary roles are also victims of systemic misclassification. Pokémon are classified as counters if their typing is super-effective against a common defensive Pokémon. However, offensive Pokémon commonly carry moves that are super- effective against defensive Pokémon but do not match their own typing. This leads to a Pokémon failing to be classified as a counter to a Pokémon it easily defeats. The same issue occurs in the opposite direction. A Pokémon may be classified as a counter to a Pokémon despite not carrying a move corresponding to the super-effective type. These issues can be corrected by creating a new object property that links a Pokémon to the types of its moves.

Table 2: Team Scores and Game Records for each team

	Gener. A	Gener. B	Gener. C	Manual
Team Score	22.000	21.083	23.166	23.250
Game Record	21-9	19-11	25-5	26-4

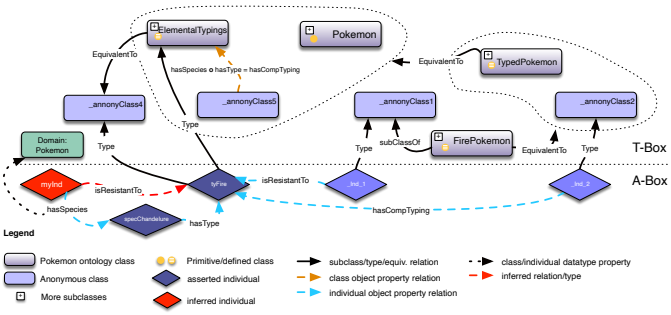

Figure 1: Inference for new individual: *isResistantTo tyFire*.

Evaluating Team Scores

The classification test has shown that in most cases, the TeamVerify tool can recognize a team's structure with over 86% precision and recall. Therefore, we can use these automated classifications as the basis to score several teams and compare their viability in play with the viability computed by TeamVerify. Four teams will be compared: one manually designed by a domain expert, and one from each of three state-of-the-art automated team generators. Their team score will be computed by TeamVerify, and each team will be used for a series of 30 games on the Pokémon Showdown battle simulator. Given the complexity of the Pokémon domain, for this experiment we expect that teams generated with simpler heuristics should score lower than more complex heuristics or our control team generated by a human expert. Similarly, we expect team score to correlate positively with the team's record in playtesting.

Generator A uses a purely statistical approach to team building (DigitalFlow 2019). Given a starting point, namely a single Pokémon, the tool scrapes usage statistics from smogon.com to fill out the team. The five most common teammates of the seed Pokémon are chosen, and all six Pokémon are given their most common moves. This tool integrates no semantic understanding and is prone to mixing strategies.

Generator B selects Pokémon from a pool of human-generated Pokémon sets (Sciarrone 2020). Given a seed Pokémon, the tool selects five teammates randomly from the pool. This tool integrates basic semantic understanding by ensuring that each individual Pokémon is viable.

Generator C also selects Pokémon from a pool of human-generated sets, but the team composition is subject to a handful of rule-based constraints (Pyrotoz 2020). The tool integrates slightly more semantic understanding because it rejects teams with certain basic deficiencies. If Pokémon Y requires support from Pokémon X to be most effective, the tool will not select Y without also selecting X.

Each of these three generators was given the same seed Pokémon, and the teams created by each generator, along with a team built by a human domain expert, were scored using the TeamVerify tool. To test the strength of the scoring scheme, each team was used for a 30 game sample on the Pokémon Showdown online matchmaker. The results of these games, along with team scores, are shown in Table 2.

The results of the experiments show that the scoring scheme used by TeamVerify is valid. Our initial hypothesis that Generator A would create the weakest team was false, given its 70% winrate. However, this reality was evident in the TeamVerify score given to Generator A. Each team's score roughly correlates to the eventual winrate. There are issues of scale present in the data. Records are skewed slightly higher due to the fact that random games are less competitive than official tournament matches, but these conditions are consistent among all four data samples. All four team scores are also within a 2.5 point range, since none are particularly unviable. These issues ultimately only affect the ease of analysis and have no effect on the results themselves.

Inspecting the results in Table 2 shows a division between two groups of two teams. The manually built team and Team C are similar in both team score and record. These correspond to the most competitively viable teams. Both have multiple win conditions and a strong defensive core to provide stability. Each Pokémon on either team is versatile, providing an average of approximately 3.8 points from its roles. Most importantly, the roles that TeamVerify measures are seemingly the factors that make these teams superior.

That can be further seen by examining Teams A and B, which score lower and have worse winrates. The two main ways to reduce a team's score are including Pokémon with few roles and including too many redundancies in an individual role. The main problem with Team B was the second factor. Though Generator B suggested three defensive Walls, all three were counters to similar typings. This redundancy left the team incredibly weak to certain offensive Pokémon that were able to defeat all three defensive Walls. Team A suffered from a lack of versatility among its Pokémon, the first factor. The team had only one win condition, which is very apparent to an opposing player. Any team with only one win condition is easy to defeat, as it is incredibly one-dimensional. These weaknesses seem to be calibrated well in the scoring scheme used by TeamVerify.

Conclusion

We have proposed the TeamVerify tool for determining whether a Pokémon team is competitively viable. The tool is based on a Pokémon ontology that allows for a more robust system than could be built using a database. By hard-coding a handful of logical axioms about the competitive Pokémon domain, a standard DL reasoner can infer properties of any user-submitted data. This compares favorably to a database where all information must be encoded and checked for viability before a user submits their team. A knowledge base with logical axioms rather than inserted facts allows the tool to continue working even with the shifting strategies in a competitive domain. A database system would require constant updates to its knowledge base.

The TeamVerify tool represents the foundation of an academic approach to strategic decision making in competitive Pokémon. The tool provides several benefits to players even without the capability to generate a team from scratch. The representation of a team through membership lists allows players to view their own teams from a more strategic perspective than other team visualization tools. Certain key concepts, such as the need for versatile Pokémon in order to

respond to a multitude of opponent strategies, become apparent through the membership list format. The team score feature allows players to get a quantitative measure of their team's viability and the effects of adjustments they make. Typically, players would need to test a team by playing games against random opponents. Less experienced players may not come to conclusions quickly from testing, and even experienced players will only have a qualitative judgement. The feature set of TeamVerify, therefore, provides value to players and acts as the foundation for a compelling area of research in esports.

Future Work

The TeamVerify tool is the first step toward rigorous analysis of strategic decision making in competitive Pokémon. With this base there are several directions for future work in the area. As shown in the evaluation of TeamVerify's automatic annotations, there are weaknesses in the decision boundary between several roles. Therefore, effort should be made to define the decision boundaries more rigorously, potentially using a neural network approach over a manual approach. The data assumption that limits the number of Pokémon can be relaxed if a more efficient data structure is used. Future work should consider pre-inferring information and indexing it into a graph database. This may lead to faster reasoning times and the capacity to consider more species of Pokémon.

The evaluation of team scores has demonstrated that a tool built on top of an ontology-based knowledge system can verify the competitive viability of a competitive Pokémon team. With this foundation, future work should also look to determine how TeamVerify can be used as the basis for a generative team building tool like the three generators compared in the previous section. Such work may look to integrate additional information from a combination of user-generated and automatically-generated team building sources. New challenges will arise in information storage and retrieval, as any information added to the ontology will slow down the reasoning process. Segmenting data that is not involved in reasoning tasks will provide another semantic challenge. However, the eventual result will be a generative model that can mimic human team building at least as well as current heuristic models.

References

[Baader et al. 2003] Baader, F.; Calvanese, D.; McGuinness, D.; Patel-Schneider, P.; Nardi, D.; et al. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge University Press.

[DigitalFlow 2019] DigitalFlow. 2019. Programming - Smogon Team Assistant. [Online; accessed 13-Oct-2020].

[Glimm et al. 2014] Glimm, B.; Horrocks, I.; Motik, B.; Stoilos, G.; and Wang, Z. 2014. HermiT: an OWL 2 reasoner. *Journal of Automated Reasoning* 53(3):245–269.

[Huang and Lee 2019] Huang, D., and Lee, S. 2019. A Self-Play Policy Optimization Approach to Battling Pokémon. In *2019 IEEE Conference on Games (CoG)*, 1–4. IEEE.

[Lamy 2017] Lamy, J.-B. 2017. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine* 80:11–28.

[Niaraki and Kim 2009] Niaraki, A. S., and Kim, K. 2009. Ontology based personalized route planning system using a multi-criteria decision making approach. *Expert Systems with Applications* 36(2):2250–2259.

[Noy et al. 2001] Noy, N. F.; Sintek, M.; Decker, S.; Crubézy, M.; Fergerson, R. W.; and Musen, M. A. 2001. Creating semantic web contents with Protégé-2000. *IEEE Intelligent Systems* 16(2):60–71.

[Plessers and De Troyer 2005] Plessers, P., and De Troyer, O. 2005. Ontology Change Detection Using a Version Log. In Gil, Y.; Motta, E.; Benjamins, V.; and Musen, M., eds., *The Semantic Web – International Semantic Web Conference (ISWC)*, volume 3729, 578–592. Springer.

[Pyrotoz 2020] Pyrotoz. 2020. Random Pokémon Generator, Pyrotoz's Pokémon Teambuilder. Webpage. Last accessed 13-Oct-2020.

[Sciarrone 2020] Sciarrone, J. 2020. Team Generator. Webpage. Last accessed 13-Oct-2020.

[Shearer, Motik, and Horrocks 2008] Shearer, R.; Motik, B.; and Horrocks, I. 2008. HermiT: A Highly-Efficient OWL Reasoner. In Ruttenberg, A.; Sattler, U.; and Dolbear, C., eds., *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*.

[Simões et al. 2020] Simões, D.; Reis, S.; Lau, N.; and Reis, L. P. 2020. Competitive Deep Reinforcement Learning over a Pokémon Battling Simulator. In *2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 40–45.

[Smogon community 2020a] Smogon community. 2020a. An introduction to smogon's tier system. https://www.smogon.com/bw/articles/bw_tiers.

[Smogon community 2020b] Smogon community. 2020b. Smogon strategy dex. https://www.smogon.com/dex/ss/pokemon/.

[Smogon community 2020c] Smogon community. 2020c. Tiering for Generation 8 - Smogon Forum Announcements. https://www.smogon.com/forums/threads/tiering-for-generation-8.3657121/.

[The Academy community 2020] The Academy community. 2020. The Academy Forums. https://discord.com/invite/nykYGrN.

[The Pokémon Company 2020] The Pokémon Company. 2020. Pokémon in Figures. https://corporate.pokemon.co.jp/en/aboutus/figures/.

[Toledo et al. 2011] Toledo, C. M.; Ale, M. A.; Chiotti, O.; and Galli, M. R. 2011. An ontology-driven document retrieval strategy for organizational knowledge management systems. *Electronic Notes in Theoretical Computer Science* 281:21–34. Proceedings of the 2011 Latin American Conference in Informatics (CLEI).