Reverse Dictionary for Video Games: Weakly Supervised Natural Language Processing

Kaitlyn Chan

Abstract

Natural language processing (NLP) is a rapidly developing field with recent applications including weakly supervised text classification that uses machine learning techniques to label documents with minimal human intervention.

As the opposite of conventional dictionaries of key-value pairs, reverse dictionaries return the key when given a value. Reverse dictionaries can be augmented using NLP models.

Video game franchises can have over a thousand characters, each with distinct characteristics. For returning fans, one may run into a tip-of-the-tongue situation in which they may recall certain facts but be unable to name a character. We provide a reverse dictionary tool for people to query such facts and receive the name of the character they were describing.

The code repository can be found at https://github.com/k6chan/reverse-dictionary-pokemon.

Introduction

Rather than the conventional dictionary structure that returns a value when given a key, reverse dictionaries return a key given the value. These have helpful applications for classification when there are many possible classes. Example applications include identifying body parts and medical conditions based on symptoms (Table 1). In this paper we apply a reverse dictionary to a video game franchise. Collectible monster series like *Pokémon* can have hundreds of characters, each with distinct characteristics, making a reverse dictionary a helpful way to query specific Pokémon based on their appearance and statistics from the games.

Query	Output
Freezing of the skin	Frostbite

Table 1: An example of a reverse dictionary's input and output.

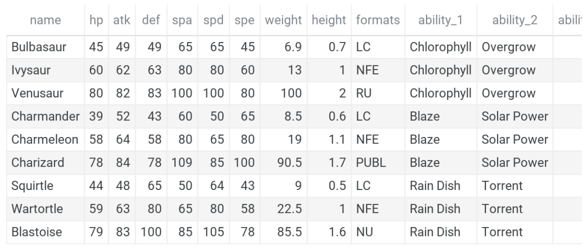
Reverse dictionaries are simple to implement if the value exactly matches or contains the dictionary definition, but in reality a user may not provide such a perfect match. Natural language processing (NLP) allows for a more realistic model to be implemented where the user's description can be used to search for the top most likely names. A baseline method that does not require human supervision is Term-Frequency-Inverse-Document Frequency when used for information retrieval (IR-TF-IDF). When adapted for text classification, IR-TD-IDF aggregates the TF-IDF values of its seed words–in this context, the user's query–as they appear in a given document and assigns a label based on the class with the greatest aggregate TF-IDF. Each document corresponds to a Pokémon.

Similar to a search engine, this reverse dictionary will provide the names of the top Pokémon with the most similarity to a user-provided query. It is available as a web app that can be hosted locally on user's computers through source code or environment on GitHub and Docker.

Pokémon data from the reverse dictionary is web scraped from the Smogon University and Bulbapedia fan wikis and aggregated into descriptions for each Pokémon that will be matched against the user-provided query.

Methods

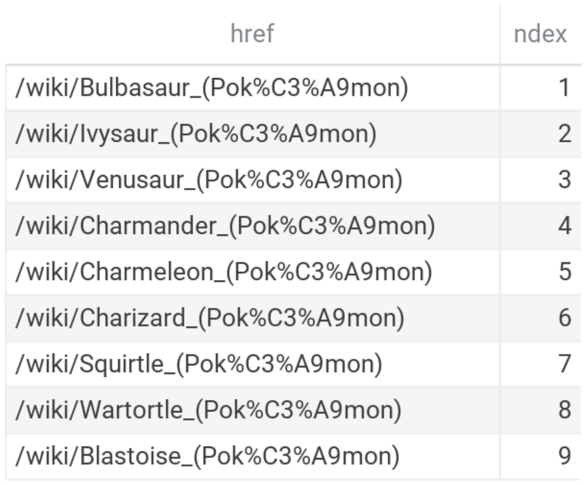
The *Pokémon* franchise is well-documented online on fan-maintained wikis. Data is first scraped from Smogon University for a “dex” identifier, numerical stats, types, abilities, competitive ranking, and immediate evolution for each Pokémon up to Generation 7 (2017) (Table 2). Exact numbers are unlikely to be recalled by users, so thresholds are set to determine if a Pokémon is “fast” or “heavy”.



name	hp	atk	def	spa	spd	spe	weight	height	formats	ability_1	ability_2	abili
Bulbasaur	45	49	49	65	65	45	6.9	0.7	LC	Chlorophyll	Overgrow	
Ivysaur	60	62	63	80	80	60	13	1	NFE	Chlorophyll	Overgrow	
Venusaur	80	82	83	100	100	80	100	2	RU	Chlorophyll	Overgrow	
Charmander	39	52	43	60	50	65	8.5	0.6	LC	Blaze	Solar Power	
Charmeleon	58	64	58	80	65	80	19	1.1	NFE	Blaze	Solar Power	
Charizard	78	84	78	109	85	100	90.5	1.7	PUBL	Blaze	Solar Power	
Squirtle	44	48	65	50	64	43	9	0.5	LC	Rain Dish	Torrent	
Wartortle	59	63	80	65	80	58	22.5	1	NFE	Rain Dish	Torrent	
Blastoise	79	83	100	85	105	78	85.5	1.6	NU	Rain Dish	Torrent	

Table 2: Web-scraped stats, competitive ranking, and abilities of Pokémon from Smogon.

Biological descriptions must be collected from another source: Bulbapedia is then scraped for a list of all Pokémon by dex number, which can allow merging onto the Smogon dataset to aggregate Pokémon data into a single table (Table 3).



href	ndex
/wiki/Bulbasaur_(Pok%C3%A9mon)	1
/wiki/Ivysaur_(Pok%C3%A9mon)	2
/wiki/Venusaur_(Pok%C3%A9mon)	3
/wiki/Charmander_(Pok%C3%A9mon)	4
/wiki/Charmeleon_(Pok%C3%A9mon)	5
/wiki/Charizard_(Pok%C3%A9mon)	6
/wiki/Squirtle_(Pok%C3%A9mon)	7
/wiki/Wartortle_(Pok%C3%A9mon)	8
/wiki/Blastoise_(Pok%C3%A9mon)	9

Table 3: Web-scraped page links for each Pokémon and their dex number from Bulbapedia.

The list of page links for each Pokémon is used to access that Pokémon’s wiki page on Bulbapedia, from where information including biological descriptions of the Pokémon can be scraped and extracted for nouns using the spaCy natural language processing library. These nouns are recorded and appended to the Smogon data.

Both the collected data and the user’s query is lemmatized and removed of stopwords. When sufficient data is compiled, all descriptors of a Pokémon are concatenated into a single paragraph with which TF-IDF can be calculated from a user-provided query. The top 5 matching Pokémon are sorted and displayed in the web app based on the cosine similarity to the query’s TF-IDF vector. This baseline method does not take sentence structure into account.

Word2Vec creates word vector representations of all the words in a corpus, creates label representations by aggregating the vectors of seed words only, then assigns all other documents a label based on the largest cosine similarity between the chosen document's vector with each label's vector (Shang). Doc2Vec is a similar tool that creates vector representations of entire documents rather than by the words.

The next model available is an extension of IR-TF-IDF that uses words similar to the user's query as suggested by a Word2Vec model pre-trained on a Wikipedia dataset.[1] The Word2Vec model supplies the top 10 most similar words to each word in the user's query. All similar words and the user's query itself is then fed into the baseline IR-TF-IDF model.

The GPT-3 Davinci API was implemented into the web app, using the user's query to ask the neural network for the top 5 Pokémon that match the query. GPT-3 uses its own collection of data sources, including Wikipedia and Common Crawl, essentially an archived version of the Web (Brown et al. 3-4). It is able to parse the query at a high level and use it to search the specific domain of Pokémon and return a sensible result. As the GPT-3 API requires credit to be used pass a trial period, it is not available on the web app.

Results

As IR-TF-IDF relies on counting and matching the exact words from the user's query to each document, it will fail to return any similar Pokémon when the user's query does not contain words found in any document (Table 4).

Bulbasaur	
Petilil	
Basculin - Blue	
Basculin	
Sandile	

Table 4: The default top 5 results provided by IR-TF-IDF on the web app when no terms from the user's query match any documents.

A standalone Word2Vec model that returned the top most similar Pokémon vector representations compared via cosine similarity to the user's query as a vector did not return Pokémon that even visually matched the query, likely due to the narrow domain and insufficient collected data for a model to be trained on or transform. Using the pre-trained model did not show a noticeable improvement to this observation. Instead, the pre-trained model was used to provide additional words similar to the user's query that are also fed into the IR-TF-IDF model, reducing the amount of times the default IR-TF-IDF result appears from Table 4. This addition of Word2Vec does help in situations where a very similar word to the user's query is in the collected data, but there are also situations where the suggestions provided by Word2Vec are unrelated to the user's query given the narrow domain; for example, Word2Vec suggests music-themed words when given the word “rock”, despite the word’s use in Pokémon usually relating to the natural object. And if the user provides a word not in the trained model’s vocabulary–which can be common since there are characters and terminology that are not common English words–the model is unable to infer its vector and throws an error.

GPT-3 showed impressive and reasonable results after fine-tuning the question the web app asks the neural network. The most complex GPT-3 model, Davinci, can *“name the top 5 actual Pokémon up to Generation 7 that match the query”* provided by the user on the web app. The word “actual” has to be included in the question to avoid nonexistent Pokémon being suggested, and “up to Generation 7” prevents Pokémon from further generations being returned. A comparison between the baseline IR-TF-IDF, IR-TF-IDF with Word2Vec, and GPT-3 models is provided (Table 5). When given a query whose words do not exist in the collected data (“octopus”), IR-TF-IDF provides the default result, IR-TF-IDF with Word2Vec does not return an octopus Pokémon but instead includes a squid and starfish Pokémon, and GPT-3 with its outside data sources is finally able to return an octopus Pokémon.

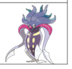
IR-TF-IDF	IR-TF-IDF with Word2Vec	GPT-3
Bulbasaur	Garchomp	Octillery
Petilil	Inkay	Tentacool
Basculin-Blue	Malamar	Tentacruel
Basculin	Starmie	Inkay
Sandile	Trevenant	Malamar 

Table 5: Top 5 results from three models (IR-TF-IDF, IR-TF-IDF with similar words suggested by Word2Vec, and GPT-3) when given the query “octopus”, which is a word not in the collected data.

An extension to the project investigating Pokémon as vectors with the collected data used Doc2Vec to convert each Pokémon’s document into a vector. Due to the same reasons behind Word2Vec’s inaccuracy, Pokémon that look visually similar or have similar theming do not always have the most similar vectors (Figure 1). For example, the golem Pokémon Regice has a closer similarity to the airplane-like Latios over another golem, Regirock.

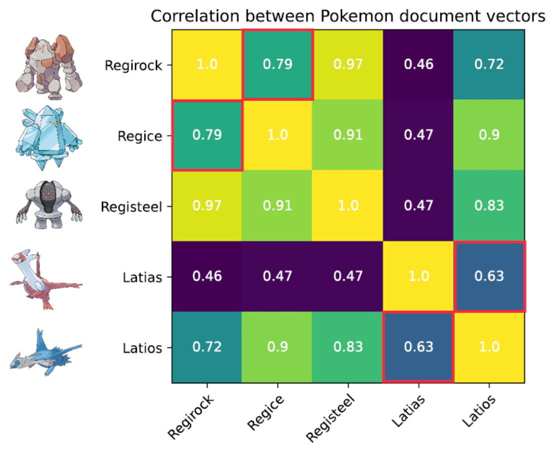


Figure 1: A heatmap of the Pearson correlation coefficient between 5 Pokémon represented as vectors. A coefficient closer to 1 means a higher linear correlation between the vectors. Red outlines describe a comparatively low correlation despite being biologically similar to the other Pokémon.

Conclusion

A reverse dictionary for a particular domain such as a video game series is an engaging method for people to reconnect with a series they enjoy. It is useful for solving tip-of-the-tongue problems where only a few descriptors of a character are known, as it can return numerous possible matches sorted by confidence.

The data collected is sufficient for baseline models and some neural networks to provide accurate results, however the latter requires more robust descriptions of each character from a variety of sources in order to cover as much of the narrow domain as possible. Extensions to this paper include additional models on the web app, more data collection from Bulbapedia such as behavior and role in the story or games. GPT-3 can be asked to provide additional descriptors on each Pokémon. Other NLP models will also benefit from sentence structure being considered and contained within the data. Inspiration for other models can be found on Hugging Face's question-answer, sentence similarity, or transformer repositories, and other research papers on reverse dictionaries like WantWords[2].

References

Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

Shang, Jingbo. "2022-Fall-DSC180B14-Capstone: Weakly Supervised NLP." Jingbo Shang, 9 Mar. 2023, https://shangjingbo1226.github.io/teaching/2022-fall-DSC180B14-capstone.

[1] *glove-wiki-gigaword-50 on* https://github.com/RaRe-Technologies/gensim-data

[2] https://aclanthology.org/2020.emnlp-demos.23/