Video Game Experience Prediction

Matthew Mulholland **Emily Olshefski** **Janette Martinez**
Montclair State University
Montclair, NJ, USA
[mulhollandm2, olshefskie1, martinezj7]@montclair.edu

Abstract

Treating number of hours played as a proxy for video game experience, we scrape review/hours played data from the Steam video game (http://store.steampowered.com/) website in order to build a corpus and eventually an NLP system for predicting experience from review text. We collect data from over 10 games and over 250,000 reviews. The foundations for our system, which is based on modern statistical NLP methods, are described and results form initial, cursory explorations of the data are provided. We believe that our results, while signalling substantial room for improvement, nonetheless suggest that there is potential for an effective experience prediction system.

1 Introduction

Ground truth in the realm of deception detection is information that can be verified or denied based on verbal or written data. In this paper we look to the hours played by the reviewer as the ground truth. The data we are basing the ground truth upon are reviews of games published on the online video game platform Steam. The ground truth in our case is the hours played values, which are provided by Steam for all reviews and which are derived from the players' own online game playing statistics (i.e., not on user-reported values). The information gathered by Steam is a good example of ”external” ground truth, along with the fact that this is real-world data. The idea is that the higher the hours played values are, the more trustworthy the reviews are. Naturally, if a reviewer has played more hours of a game, then he/she has a better understanding of the game and, in turn, can write a more nuanced and accurate review of a game or a review that exhibits information that can only be acquired through usage.

Data was collected from the video game platform Steam, available or PC, Mac, and Linux, due its popularity as a gaming platofrm as well as availability of data. We developed a web-scraping method in order to build a corpus of reviews. Reviews from the top 11 most popular games were scraped from the Steam website. Number of hours played were also collected in conjunction with the review text associated with them. After pruning the data – filtering out non-English reviews, reviews that had close to no content, etc., training/test set partitions were built. Using Weka and a basic bag-of-words approach in order to get some initial results, we built regression models with the SMOreg machine learning algorithm. This work is the first steps in using novel and more comprehensive data not otherwise used in similar studies.

2 Related Works

While this paper does not fall under the grounds of typical deception detection work, influences from other deception detection work form the basis for some of the overarching ideas behind this research. Reviews in this paper are based on trustworthiness.

Jindal and Liu (2008) used product review data from `Amazon.com` to find opinion spam. Discovering opinion spam led to studies in researching deceptive reviews. Since there was no gold standard data to work from at that time, they initiated detection of such spam by first detecting duplicate reviews, followed by using supervised learning with manually labeled training examples from a second type of spam consisting of opinions not mentioning the product specifically, and a third type with irrelevant content (such as ads and random texts.) If the review did not fall in these domains, the review was considered truthful. They had success with logistic regression. Each `Amazon.com` review consists of 8 parts: Product ID, Reviewer ID, Rating, Date, Review Title, Review Body, Number of Helpful Feedbacks, and Number of Feedbacks.They gathered reviews from different categories such as music, books, dvds, and mProducts (consisting of industry manufactured products.) Software and video games were not specifically mentioned. `Amazon.com` reviews were also used by Fornaciari and Poesio (2014) to test identification of fake Amazon reviews using the learning from crowds method.

Ott et al. (2011) developed gold standard deception review data containing 400 true and 400 false reviews. 89.6% accuracy was achieved using BIGRAMS. The 400 fake reviews were written by Amazon Mechanical Turkers. The real ones were mined from `Trip Advisor.com` (and pruned to match the criteria given to the turkers). Naive Bayes, Support Vector Machines(SVM LIGHT), and the Standford Parser were used.

Feng et al.(2012a) Used Otts gold-standard corpus, along with a heuristic method and PCFG parse trees for deep syntax encoding were favored to shallow syntactic features (referring to LIWC and pos tags, however they were used to garner comparisons of effectiveness). They used support vector machine liblinear and improved Otts accuracy score to 91.2% by way of deep syntax and bigrams.

One advantage of using data from Steam (as opposed to `Amazon.com` or `TripAdvisor.com`, both of which have stored tranasctional data) is that Steam's data goes a step beyond the ground truth of the aforementioned websites. Not only is transactional data easily accessible, but the fact is that hours played values represent ground truth that a player truly has experience with the product. Furthermore, transactional data is not even totally necessary to determine ground truth since Steam allows its users to play games owned by other Steam users on their friends list and records the hours played. Therefore, Steam users don't need to necessarily own a game in order to write a truthworthy review. Regardless, there is no litmus for the prior works to test trustworthiness in the same way that Steam does, hence making it a useful platform to collect review data.

3 Methods

3.1 Data Collection and Selection

As mentioned previously, Steam was chosen as the platform to obtain data due to its popularity as well as the availbility of solid ground truth data in terms of hours played as well as review text. Originally we chose the top 10 most played games on Steam via `http://store.steampowered.com/stats` (highest peak number of current players in 24 hours). This list included the following games: Arma 3, Counter Strike, Counter Strike: Global Offensive, Dota 2, Football Manager 2015, Garry's Mod, Sid Meier's Civilization 5, Team Fortress 2, The Elder Scrolls V: Skyrim, and Warframe. The idea behind choosing the 10 most popular games is that these games would have a large number of reviews due to their popularity. Like Fornaciari and Poesio (2014), who used reviews of books by well-established authors (like Stephen King and Rudyard Kipling) on A`Amazon.com`, many of the games used in this work are versions in classic video game series (most notably the Arma, Counter Strike, and Grand Theft Auto series). Also, these games are consistently popular on Steam, as seen from monitoring the aforementioned website.

The decision to add another game, Grand Theft Auto V came from the notion that, while Grand Theft Auto V was available on other non-PC platforms (the game consoles Xbox 360, Xbox One, and Playstation 3 and 4), it had a highly anticipated release date on April 14, 2015. The data from this game was not only included to widen the scope of genres analyzed but also to account for games that players might have extensively played on a different platform.

Originally, our intention was to use the Steam API since it allows anyone with a Steam account to collect a multitude of data-points on Steam users and games. However, review text was not available in the Steam API and HTML scraping of the Steam website for each games was implemented in order to extract review text and hours played. Using Python, a tool was developed that was able to successfully scrape review/hours played data from the Steam website. In little more than one week, over

250,000 reviews were collected. After the data was scraped, it was filtered and pruned so that it could be added to a corpus of game reviews.

3.2 Data Pruning

We used real world data, directly scraped from Steam reviews of the eleven aforemetioned games, and were able to collect many more reviews than we could possibly process with our current computing resources. However, we overestimated the number of reviews we would need because we anticipated that there would be occasional empty reviews, reviews in languages other than English, etc. We filtered the reviews and processed them with another script written in Python, which categorized the reviews and transformed the hours played values into a ten-point scale. All reviews were inserted into a MongoDB database so that they could be accessed and processed with ease. Filtering consisted of eliminating non-English reviews and throwing out reviews whose hours played values were greater than two standard deviations above the mean (within set of game reviews) or that had less than 50 characters of content. We set the minimum number of hours played to zero since we wanted to be able to see examples of reviews whose authors presumably had little experience of that which they were reviewing. One reason we wanted to standardize the reviews in this way was because the distribution of review hours was so skewed. Usually, up to 90% of the data could be accounted for by one-fifth of the distribution. In other words, if what we planned to predict was more-or-less on a scale of 0 to 1,000, then we did not want reviews whose authors had 5,000 hours played since the difference between 1,000 and 5,000 is less significant than the difference between 500 and 1,000. Further, the conversion script broke the hours played values down into a 1-10 scale so that we could possibly train on multiple games at once and then use the resulting model to test on still different games. For example, in a given case hours played values 0 to 100 could be converted to 1 (which could be thought of as the least amount of experience), 101 to 200 could be converted to 2 (very little experience), etc. This way, the prediction would really only have about 10 labels (that happened to be on a scale).

We took further preprocessing steps in order to get the final review text: lower-casing, tokenization with the NLTK python library, a series of hand-crafted rules for converting improper contractions to their full representations, etc. After processing, we ended up with 121,434 reviews, just under 50% of the total number of reviews. A discussion of our proposed system is given below and some initial discussion of results is provided.

4 Proposed System

Our initial plan was to create an end-to-end NLP system that would be able to take review text as input and then output a prediction for the experience of the author who wrote the review. We finished creating a system that could extract features and train a machine learning model, but we were unable to complete the other half of the proposed system in time for this paper. However, we would like to take some space here to describe our methodology.

In terms of textual features, we decided to avoid using LIWC-style features, unlike in similar studies in the past, as we are inclined more towards statistical natural language processing. As such, we of course were going to make use of the standard bag-of-words approach. However, instead of using only unigrams, we decided to extract bigrams as well. Furthermore, instead of using only token *n*-grams, we decided to make use of character *n*-grams for *n* = 2 through 5 since we felt that there could be an abnormally high importance of character sequences (for example, in emoticons or URLs). Lastly, we decided to extract syntactic/semantic depedency features. In order to do this, we used the spaCy Python library for NLP-related tasks. spaCy's API provides a convenient way to do tokenization, part-of-speech tagging, and parsing. For the dependency features, we simply extracted any depedency relationship between two non-punctuation tokens of the review text. For example, ”one step forward” would result in two depedency features: ”step:one” and ”step:forward” as ”step” is the head in both cases. For an example of a review's database representation (after feature extraction), see the appendix.

Our proposed system communicates with the MongoDB database to find training set reviews for a given game, extracts features to create frequency distributions of phenomena, and then binarizes all features and creates a sparse file format, which can be used with SciKit-Learn and SKLL (SciKit-Learn Laboratory) machine learning Python libraries. Our system then runs a model building process with the training data and stores the resulting model as a picklable object, i.e., a programming object that can be transferred via files. By default, we use the support vector regression algorithm and we tune to algorithm's parameters so as to optimize quadratic weighted kappa, i.e., how well the resulting system can predict the 1-10 scale values. However, many other machine learning algorithms and objective (i.e., tuning) functions can be used. For example, RandomForestRegressor, DecisionTree, etc., for examples of other supported algorithms, and off-by-one quadratic weighted kappa, *r*2, and others, for the supported objective functions.

5 Results & Discussion

Since our prediction/evaluation system was unfinished, we instead decided to write a Python program to automatically generate ARFF files containing the review text and hours played values alone. ARFF files are supported by Weka and, therefore, we believed we could perhaps get a rough estimation of how our system might perform when finished. (However, it must be emphasized that ARFF does not support sparse file formats and, thus, we could only hope to make use of Weka functionality to turn the review texts into word vectors, i.e., make use of the bag-of-words approach. Testing out a few games only (and on only a handful of algorithms), we were only able to attain a maximum correlation coefficient of 0.30. As we said earlier, we believe that this is both lower than we can hope to attain with the full system and a sign that there is a possibility for improvement. If the unigram representation of the reviews itself can lead to a performance of up to 0.3, then perhaps the use of many more textual features could result in a much higher ability to effectively predict reviewer experience. Due to physical memory limitations, we were limited as to how many tests we could do on the data with the available classifiers. Linear Regression alone took more than one hour on one data-set (and we have 11 games). We chose to run our data-set through SMOReg as it processed the fastest and is closest to our developed system. The SMOReg classifier implemented support vector machines for regression and we used the RegSMOImproved algorithm, specifically.

6 Future Work

In the future, we would like to explore the possibility of applying this method to data that lacks ground truth, and perhaps outside the genre of video games. For example, what would be the equivalent of hours played in the domain of hotel reviews, restaurant reviews, etc.? `TripAdvisor` has sub-categories for their reviews (”cleanliness”, for example), but that is left to the discretion of the reviewer, while hours played values cannot be manipulated. While the former is completely subjective, hours played is objective and immutable.

As Jindal and Liu (2008) used helpful reviewer feedback, we would work on fusing other sources as factors for ground truth, including bans and achievements obtained (and, consequently, difficulty of achievements obtained) to gauge the reviewers' competence. Further, the Steam API features playtime_2weeks, the total number of hours played in the last 2 weeks, all of which can be obtained through the Steam API without any additional scraping. This information would be helpful to us in further developing this work.

In the future we would also like to include more representative games and more domains in general. We would also like to apply this method to review data on respected/popular gaming review sites, like `metacritic.com`, `ign.com`, and `pcgamer.com` where reviewers have a monetary incentive when writing reviews.

GitHub Repository

https://github.com/mulhod/reviewer_experience_prediction.git

Bibliograpy

Nihar Jindal and Bing Liu. 2008. Opinion spam and analysis. *Proceeding of the International Conference on Web Search and Web Data Mining* , 219-230. ACM.

Tommaso Fornaciari an Massimo Poesio. 2014. Using Web-Intelligence for Excavating the Emerging Meaning of Target-Concepts. *Proceedings of the*

14th Conference of the European Chapter of the Association for Computational Linguistics, 279-287. Gothenburg, Sweden. April. Association for Computational Linguistics.

Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 2011. Finding deceptive opinion spam by any stretch of the imagination *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 309-319. Portland, Oregon. June. Association for Computational Linguistics.

Song Feng, Ritwik Banerjee, and Yejin Choi 2012a. Syntactic stylometry for deception detection. *Pr Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 171-175. Jeju Island, Korea. Association for Computational Lingusitics.

Appendices

See attachment.