NLP & Machine Learning Applied: Video Game Reviews

Matthew D. Mulholland
Montclair State University
Montclair, NJ
mulhollandm2@montclair.edu

Abstract

Despite the ambiguity of the concept, much research has been done in the area of detecting "fake" reviews. It is, however, often difficult to build corpora containing reviews that are definitively "fake" or "true". It is simpler to reframe the issue in terms of the amount of experience a reviewer has with a product: given a review, can we tell anything about the level of experience the reviewer has with the reviewed product?

In this exploratory paper, I will detail a research project whose aim was to relate video game reviews to proxies for reviewer experience, such as number of hours played, number of times marked as helpful, and other related review/user attributes, using natural language processing and machine learning techniques. The ultimate end is to produce a capability for ranking or filtering reviews, which could be used in addition to or in place of other fake review or spam filtering algorithms.

A less grand aim of the project was to scrape review data from the Steam video game website and make it publicly available. This data will be described at length.

1 Credits

I would like to thank both Janette Martinez and Emily Olshefski for their help in the initial iteration of this work as part of a class project. They helped lay out the problem, make decisions regarding the source of the data and the games for which data was collected, and also write some of the preprocessing code. Some sections of this paper build on the final paper for that class project, which they took a lead in writing.

2 Introduction

2.1 Reframing a familiar problem

In the realm of deception detection, ground truth – information that can be verified or denied – is not easy to come by. For example, one application of deception detection is in the detection of fake reviews: reviews that are known to be fake (by some external means) are compared to reviews that are believed to have been written in good faith. It may be easy in some cases to identify certain reviews as fake since the authors themselves might admit as much. However, the categorization of other reviews from either category is a difficult matter. Thus, corpora of fake/real reviews are often constrained in that, for any given review, it could be impossible to determine the category.

In this paper, I propose a fundamental reframing of the issue: the problem should not be about whether or not a given review is fake, but rather about measuring the amount of experience a reviewer has with the product being reviewed. Reviews for products of which the reviewer has little to no experience – whether they have been written in bad faith or simply from a relatively uninformed perspective – could be distinguished from reviews for which the reviewer does have experience with the product being reviewed. Further, different levels of experience – say, moderate or high – could be distinguished from one another.

A system that could accurately predict the amount of experience a reviewer has given only the review text or some combination of the review text and other attributes of the review/reviewer, such as the number of times a review has been marked as helpful, could be useful in a production environment as a component in a review-filtering and/or -sorting algorithm.

2.2 How is experience measured?

To model reviewer experience (which is, after all, just as nebulous as the distinction between fake and true reviews), there must be methods of approximating experience. One simple way to approximate this value is by recording the amount of time a user has actually spent using a product before reviewing it. For most products, however, the feasibility of recording the actual time spent is low due to the cost and logistics of such an undertaking. And that is assuming time even can be recorded! In other cases, product use is more of a binary value: Did the consumer eat the pie or not? Did the user watch the movie or not? And, in still other cases, usage might need to be measured in terms of the number of times the product was used, such as lotion, hair spray, etc.

Aside from trying to measure a reviewer's actual usage of a product, there may exist other ways of approximating experience. One rather indirect way would be

to compare the number of times each review is marked as helpful (or not helpful). Presumably, reviews that are more helpful are reviews that are from users who actually used the product and vice versa.

2.3 Using video game reviews from Steam

In the realm of video games, the methods of approximating user experience described above are actually feasible. The Steam online video game platform (`http://store.steampowered.com/`) is used by video game enthusiasts for a variety of purposes. Steam functions as a platform

- in which users can play video games in an online, social environment
- from which users can purchase, research, and review video games
- for social iteraction (through video game reviews, marking reviews as helpful, funny, etc., displaying playing stats and achievements, etc.)

Steam keeps a record of the number of hours each user has played each game and, thus, when a user submits a review of a video game, this information is presented alongside the review. It is true that the meaning of this measure is not completely straightforward: a user could have played a particular game for many hours outside the Steam platform and these hours would not be included. However, it is a reasonable assumption to make that the majority of the values recorded by the platform will be accurate and/or that the amount of time that players have played a particular game outside the confines of the online platform will be similar across players submitting reviews for the same game.

The window into a user's experience with a product that is afforded by the amount of time a user has used a product is somewhat unique to the case of video games: for other products, it might be impossible or pointless to attempt to record the amount of time the user used the product. For example, there is no way to keep a record of the amount of time a user has spent using a vacuum short of conducting a study and painstakingly recording such information.

However, if experience could successfully be modelled in the case of video games, perhaps the models could be generalized to cover whole categories of products and, thus, the situation here would apply to much more than video games. Furthermore, there are other indications of a user's experience with a product, as mentioned above, such as the number of times a review has been marked as helpful, and fortunately Steam also keeps a record of such information.

2.4 Description of research

In this paper, I explore the amount of time a reviewer spent playing the game he/she is reviewing, the number of times it was marked helpful by review readers, and other attributes of the review in relation to review text itself (and also to a set of attributes about the review/reviewer, such as the reviewer's number of friends, the number of times the review was marked as funny, etc.). The greater the amount of experience a reviewer has with the game being reviewed, the greater the likelihood is that the review produced is trustworthy or in some sense valuable. Naturally, if a reviewer has spent a lot of time playing a game or if the review is voted to be relatively helpful, etc., then the review has a higher chance of representing a good understanding of the game on the part of the reviewer.

Fortunately, the online gaming platform Steam collects such data about its users and makes it publicly accessible. A web-scraping method was developed and used to build a corpus of review texts along with a lot of additional data, as mentioned briefly above. Reviews from 11 of the most popular games were scraped from the Steam website. After filtering out non-English reviews, the data was partitioned into training/test sets. A number of commonly-used NLP feature types were extracted from the reviews and machine learning experiments were conducted using a range of learning algorithms.

The main motivation of the machine learning experiments was to determine whether or not the various proxies for experience under consideration demonstrated potential in terms of whether or not they could successfully be modelled. The results are mixed, but there is some indication that this is a worthy research effort and that it could lead down the road to some of the grand aims mentioned above, i.e., influencing a review filtering/sorting algorithm.

3 Related Works

While this paper does not fall under the purview of typical deception detection work, influences from other deception detection work form the basis for some of the underlying ideas of this research. (Jindal and Liu, 2008) used product review data from `Amazon.com` to find "opinion spam". Discovering opinion spam led to research into 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 and then by using supervised learning with manually-labeled training examples. They gathered reviews from different categories such as music, books, and DVDs, but video games were not specifically included. Amazon reviews were also used by (Fornaciari and Poesio, 2014) to test identification of fake reviews using crowdsourcing.

(Ott et al., 2011) developed gold standard deception review data consisting of 400 true and 400 false reviews. The 400 fake reviews were written by Amazon Mechanical Turkers while 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 and an accuracy of 89.6%

was achieved using bigram features. (Feng et al., 2012a), using Ott's gold-standard corpus, extracted PCFG parse trees for deep syntax encoding along with some shallower syntactic/semantic features (such as LIWC features and POS tags to garner comparisons of effectiveness) in addition to the original bigram features. With support vector machine, they were able to improve on Ott's accuracy score, increasing it to 91.2%.

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 provide a window to the ground truth of product experience. There is no practical litmus test for the prior works to test trustworthiness in the same way. Rather, prior works have had to resort to circuitous ways of testing the trustworthiness of reviews.

4 Data and Code

4.1 Data

Data from 11 popular video games was scraped from the Steam website via a web scraping program written in Python. See Table 1 for a full listing of the games and the number of reviews that were collected for each game (after filtering of non-English reviews, reviews that had zero content, etc.). The program was designed to scrape the following (non-exhaustive) types of information from reviews:

- review text
- amount of time reviewer spent playing game (in hours and to one decimal place)
- number of times a review was marked helpful
- number of times a review was marked unhelpful
- number of times a review was marked funny
- number of comments other users have left in the review's discussion space
- number of friends the reviewer has on the Steam platform
- number of Steam groups the reviewer is a member of
- number of reviews, guides, and screenshots the reviewer has posted (for this and other games)
- number of Steam "badges" the reviewer has received
- number of in-game achievements the reviewer has attained

All data-sets have been made freely available with a permissive license (MIT) via `GitHub.com`. The data can be found at the following URL: `https://github.com/mulhod/steam_reviews`. It is stored in relatively common and portable `JSONLINES/NDJ` format. The data was collected over a period of approximately two weeks in the summer of 2015.

Game	# reviews
Arma 3	7,151
Counter Strike	6,040
Counter Strike: Global Offensive	7,073
Dota 2	9,720
Football Manager 2015	1,522
Garry's Mod	7,151
Grand Theft Auto V	13,349
Sid Meier's Civilization 5	7,467
Team Fortress 2	5,676
The Elder Scrolls V	7,165
Warframe	7,123

Table 1: Review data-sets. Shows the name of each game and how many reviews were collected.

4.2 Code

Like the data, all code written for and used in this project has been made freely available via `GitHub.com`. The code repository can be found at the following URL: `https://github.com/mulhod/reviewer_experience_prediction`. Also provided are in-depth descriptions of the data, full ReST-style documentation for the code, clear instructions on how to get set up and running, and numerous Jupyter (formerly known as IPython) notebooks describing various algorithms and aspects of the code and providing visualizations of the data, etc. All code is written in the Python programming language and the total number of lines of code exceeds 6,000. Cython is used to convert large parts of the code base to C extensions to improve performance. The `spaCy` Python library is used for most of the natural language processing (NLP) analysis.

5 Methods

After collecting reviews from the Steam website, the reviews were analyzed and a set of NLP features were extracted. All NLP features and review/reviewer attributes were stored in a MongoDB database collection. MongoDB is particularly useful in cases where the data is unstructured: it allows for easily inserting new fields for existing document entries. In the case of this project, the amount of data and the time it takes to

extract NLP features for each review text necessitated some creative thinking in terms of how best to store the data and reduce the computational load/the time needed to conduct many different experiments with potentially the same data. It was determined that a database holding all reviews and all NLP features would be the best solution despite the fact that the database would need a lot of memory devoted to it – the database ended up needing over 200 GiB of hard-drive space! The benefits of having a running database are that the data can be accessed at any time and that it can be queried in a way that would be impossible if it existed in flat files.

5.1 NLP feature extraction

Using the `spaCy` Python library for NLP analysis and other tools, all review texts were analyzed and NLP features were extracted. The following types of NLP features were extracted:

- word n-grams for n = 1 to 2
- character n-grams for n = 2 to 5
- syntactic dependency relationships
- review length ($log_2 x$ where x refers to the number of characters in the review)
- Brown corpus cluster IDs (for each word in the review text)

For the first feature type in the list above, otherwise known as "the bag-of-words" method, the review text is lower-cased and tokenized and then all one- and two-word sequences are identified. Likewise, for the character n-grams features, all two-, three-, four-, and five-character sequences are extracted from the raw, unprocessed review text. This latter feature type serves a couple different purposes:

- It extracts some textual features that would not be extracted via the n-gram features, such as emoticons, sequences of non-word (and word) characters, etc.
- It implicitly implements a basic form of automatic spelling correction as even misspelled tokens will still have large sub-strings in common with the correctly-spelled versions.

The third type of extracted feature, syntactic dependencies, extends the "bag-of-words" approach. The n-gram features represent sequences of characters/tokens and, for some simple cases (in English, at the very least), this suffices to extract a full syntactic structure. However, when syntactic structures are not sequential/linear or there is intervening content, such as embedded clauses, modifiers, etc., or the relationship simply manifests itself over a long stretch of words, the n-gram features are not enough. The syntactic dependency features are meant to cover such phenomena.

Lastly, the review length (as a base-2 logarithm of the number of characters, whose slope will decrease with review length) and a set of Brown corpus cluster IDs are extracted from each review. Although it is expected that review length may correlate weakly with experience (the more a user has used a product, the more he/she may have to say about it), the relationship is not necessarily linear. The difference between a review that is 10 characters long versus a review that is 100 characters in length is perhaps more significant than the difference between a 5,000-character review and a 10,000-character review. The Brown corpus cluster IDs are intended to capture a crude type of vocabulary distribution feature. Each token's cluster ID refers to a particular cluster of texts from the Brown corpus. Representing each review as a set of cluster IDs should provide some abstract idea about the content of the review.

All extracted NLP features (except for the length feature) are binarized, e.g., if an 2-gram such as "the game" appears five times in a review, the extracted feature is converted to 1, signifying that that particular 2-gram appeared in the review. Features that do not occur in a review have an implicit value of 0. Thus, sparse vectors are used to create a vector space model.

The feature set used here is meant to approximate a sort of baseline NLP system. The main aim is to model reviews with a baseline system and then evaluate this kind of system's performance in terms proxies for user experience, including the amount of time the user spent using the product he/she reviewed (in hours), the number of times the review was marked as helpful, etc. If a basic system shows some potential in this regard, it may be taken as a sign that further feature set development could lead to better performance and, thus, that the main problem – filtering/sorting reviews by their truthfulness, helpfulness, etc. – can successfully be modelled.

5.2 User/review attributes

A set of user/review attributes is also extracted for use in the machine learning experiments. These features can be used in isolation, in combination with the NLP features, or not at all. For a non-exhaustive list of these attributes, see "Data" (Subsection 4.1).

Some of these features might typically be available for reviews and, therefore, could potentially be used to improve the performance of the modelled relationship between reviews and the phenomena related to reviewer experience. However, part of the aim in conducting experiments with these additional features is to explore the possibility that the relationship between reviews and reviewer experience could be successfully modelled by such features in isolation, i.e., without any NLP features at all. For instance, to what degree can knowing the number of friends a reviewer has tell how trustworthy his/her review is?

The experiments described below will explore each

situation: models trained on NLP features, NLP features in combination with the review/user attributes, and the review/user attributes alone. Note that, when using review/user attributes to train a model to predict the number of times a review is marked as helpful, any review/user attributes that are related to the prediction label are automatically excluded to ensure that the model does not contain information that would definitely be related to what is actually being modelled. For example, if the prediction label is the number of times a review is marked as helpful, attributes such as the number of times a review was marked as unhelpful would be excluded.

6 Experiments

The aim of the experiments conducted as part of this exploratory work was to attempt to model reviewer experience and/or review usefulness in terms of the reviews and/or attributes of the reviewer/review.

6.1 `learn`

As part of `reviewer_experience_prediction`, the code repository that was developed for every phase of this work (which can be found at the following URL: `https://github.com/mulhod/reviewer_experience_prediction`), a utility was created called `learn`, which is used to conduct iterative machine learning experiments using `scikit-learn` under the hood. The motivation for making the experiments iterative was simply to try to reduce memory consumption: for each round of learning, only a small set of reviews, associated features, etc., must be stored in RAM at any given time. It is not unheard of for even a relatively small dataset of 1,000 reviews to require upwards of 50 GB of RAM after vectorization since the full feature set will reach into the hundreds of thousands. With the iterative learning algorithm used by `learn`, the user specifies the number of rounds/number of samples to use per round and the machine learning model is updated incrementally.

`learn` is the main entry-point of the system in terms of conducting machine learning experiments. Any set of games can be used for training (and possibly two different sets of games can be used for training/testing, a line of research that I hope to follow up on in the future). Five machine learning algorithms from `scikit-learn` are available, including `Perceptron`, `PassiveAgressiveRegressor`, `MiniBatchKMeans`, `MultiNomialNB`, and `BernoulliNB`. The aim in making this exact set of learners available was not necessarily to provide a representative sampling of popular algorithms, but rather it was due to constraints placed on the set of `scikit-learn` learners that could be used in an iterative fashion. Other aspects of the experiments that can be configured are the prediction label that is used (any of the reviewer/review attributes, e.g. number of game-hours played, number of times marked helpful, etc., see Subsection 4.1), the types of featues used in the model (NLP features and/or a set of the reviewer/review attributes that are not directly related to the prediction label), and much more, such as whether to use "feature hashing" (further reduces memory consumption, but does not allow for subsequent model introspection, i.e., if the user wants to see the various model features and their weights), whether or not to generate the performance metrics for a "baseline" majority label system to provide a basis on which to compare the results, etc.

Reviewer/review attribute values must be manipulated before the learning process in order to allow for meaningful learning. For example, the values of the number of hours played attribute for any given game can range from 0 to upwards of 15,000 and, thus, it would not be a meaningful exercise to try to learn how the features map to this scale. `learn` makes use of a method of breaking down a scale such as 0 to 15,000 into various "bins", possibly of equal size, but not necessarily so. If the "bin" factor is 1.0, then the scale will be divided up into roughly equal-sized partitions and each of those partitions will be mapped to a new value, i.e., the position of the bin, with numbering starting at 1. For example, if 0 to 15,000 is divided up into three bins and the factor is 1.0, then it will result in the following type of mapping: [0, 5,000] $\rightarrow 1, [5,001, 10,000] \rightarrow 2, [10,001, 15,000] \rightarrow 3$.

Often, however, the raw values of the reviewer/review attributes are heavily positively skewed. For example, most reviewers have played the game being reviewed for less than 1,000 hours, but the distribution will have a long tail to account for the relatively small percentage of reviewers who have spent much more time playing the game. Thus, in some cases, it would be better to be able to include only 0 to 500 hours in the first bin, 501 to 1,500 in the next, and 1,501 to infinity for the last bin. The "bin" factor can be used to try to account for the highly positively skewed distribution of prediction label values. `learn` allows the user to specify how the label values should be "binned". At the end of the process, the predictions that the system makes will have to be understood in terms of this binning process. One can think of a new 1-3 scale as mapping roughly to a low, medium, and high experiential scale, for example.

More information on the distribution of label values (for hours played, specifically) can be found in the `GitHub.com` data-sets repository, a separate repository that is used to store only the data itself, which can be found at the following URL: `https://github.com/mulhod/steam_reviews`.

6.2 "Number of hours played" experiments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulpu-

Game	System	Learner	Accuracy	Precision	QWK	QWK-1
Arma 3	majority label		0.71	0.51	n/a	n/a
Arma 3	NLP + non-NLP	MultinomialNB	0.53	0.66	0.13	0.38
Arma 3	NLP	Perceptron	0.6	0.57	-0.01	-0.06
Arma 3	non-NLP	Perceptron	0.52	0.56	-0.02	-0.08
Counter Strike	majority label		0.81	0.66	n/a	n/a
Counter Strike	NLP + non-NLP	MiniBatchKMeans	0.6	0.71	0.16	0.06
Counter Strike	NLP	MultinomialNB	0.82	0.69	0.16	0.28
Dota 2	majority label		0.5	0.25	n/a	n/a
Dota 2	NLP + non-NLP	Perceptron	0.53	0.49	0.08	-0.15
Dota 2	NLP	PassiveAgressiveRegressor	0.42	0.47	0.16	0.31

Table 2: A selection of game-hours experimental results.

tate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Game	System	Learner	Accuracy	Precision	QWK	QWK-1
Dota 2	majority label		0.93	0.86	n/a	n/a
Dota 2	NLP + non-NLP	MultinomialNB	0.96	0.96	0.65	0.65
Dota 2	NLP	Perceptron	0.96	0.97	0.77	0.78
Dota 2	non-NLP	Perceptron	0.93	0.86	0.0	0.0

Table 3: A selection of "number of times marked helpful" experimental results.

6.3 "Number of times marked helpful" experiments

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Acknowledgments

References

Bibliograpy

Nihar Jindal and Bing Liu. 2008. Opinion spam and analysis. *Proceeding of the International Confer-*

ence on Web Search and Web Data Mining, 219-230. ACM.

Tommaso Fornaciari and 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.