# 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

## 7 General Instructions

Manuscripts must be in two-column format. Exceptions to the two-column format include the title, authors' names and complete addresses, which must be centered at the top of the first page, and any full-width figures or tables (see the guidelines in Subsection 7.5). **Type single-spaced.** Start all pages directly under the top margin. See the guidelines later regarding formatting the first page. The manuscript should be printed single-sided and its length should not exceed the maximum page limit described in Section 10. Do not number the pages.

### 7.1 Electronically-available resources

We strongly prefer that you prepare your PDF files using LaTeX with the official ACL 2015 style file (acl2015.sty) and bibliography style (acl.bst). These files are available at `http://acl2015.org`. You will also find the document you are currently reading (acl2015.pdf) and its LaTeX source code (acl2015.tex) on this website.

You can alternatively use Microsoft Word to produce your PDF file. In this case, we strongly recommend the use of the Word template file (acl2015.dot) on the ACL 2015 website (`http://acl2015.org`). If you have an option, we recommend that you use the LaTeX2e version. If you will be using the Microsoft Word template, we suggest that you anonymize your source file so that the pdf produced does not retain your identity. This can be done by removing any personal information from your source document properties.

### 7.2 Format of Electronic Manuscript

For the production of the electronic manuscript you must use Adobe's Portable Document Format (PDF). PDF files are usually produced from LaTeX using the *pdflatex* command. If your version of LaTeX produces Postscript files, you can convert these into PDF using *ps2pdf* or *dvipdf*. On Windows, you can also use Adobe Distiller to generate PDF.

Please make sure that your PDF file includes all the necessary fonts (especially tree diagrams, symbols, and fonts with Asian characters). When you print or create the PDF file, there is usually an option in your printer setup to include none, all or just non-standard fonts. Please make sure that you select the option of including ALL the fonts. **Before sending it, test your PDF by printing it from a computer different from the one where it was created.** Moreover, some word processors may generate very large PDF files, where each page is rendered as an image. Such images may reproduce poorly. In this case, try alternative ways to obtain the PDF. One way on some systems is to install a driver for a postscript printer, send your document to the printer specifying "Output to a file", then convert the file to PDF.

It is of utmost importance to specify the **A4 format** (21 cm x 29.7 cm) when formatting the paper. When working with `dvips`, for instance, one should specify `-t a4`. Or using the command `\special{papersize=210mm,297mm}` in the latex preamble (directly below the `\usepackage` commands). Then using `dvipdf` and/or `pdflatex` which would make it easier for some.

Print-outs of the PDF file on A4 paper should be identical to the hardcopy version. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs as soon as possible.

### 7.3 Layout

Format manuscripts two columns to a page, in the manner these instructions are formatted. The exact dimensions for a page on A4 paper are:

- Left and right margins: 2.5 cm
- Top margin: 2.5 cm
- Bottom margin: 2.5 cm
- Column width: 7.7 cm
- Column height: 24.7 cm
- Gap between columns: 0.6 cm

Papers should not be submitted on any other paper size. If you cannot meet the above requirements about the production of your electronic submission, please contact the publication chairs above as soon as possible.

### 7.4 Fonts

For reasons of uniformity, Adobe's **Times Roman** font should be used. In LaTeX2e this is accomplished by putting

```
\usepackage{times}
\usepackage{latexsym}
```

in the preamble. If Times Roman is unavailable, use **Computer Modern Roman** (LaTeX2e's default). Note that the latter is about 10% less dense than Adobe's Times Roman font.

| Type of Text | Font Size | Style |
|---|---|---|
| paper title | 15 pt | bold |
| author names | 12 pt | bold |
| author affiliation | 12 pt | |
| the word “Abstract” | 12 pt | bold |
| section titles | 12 pt | bold |
| document text | 11 pt | |
| captions | 11 pt | |
| abstract text | 10 pt | |
| bibliography | 10 pt | |
| footnotes | 9 pt | |

Table 2: Font guide.

### 7.5 The First Page

Center the title, author’s name(s) and affiliation(s) across both columns. Do not use footnotes for affiliations. Do not include the paper ID number assigned during the submission process. Use the two-column format only when you begin the abstract.

**Title**: Place the title centered at the top of the first page, in a 15-point bold font. (For a complete guide to font sizes and styles, see Table 2) Long titles should be typed on two lines without a blank line intervening. Approximately, put the title at 2.5 cm from the top of the page, followed by a blank line, then the author’s names(s), and the affiliation on the following line. Do not use only initials for given names (middle initials are allowed). Do not format surnames in all capitals (e.g., use “Schlangen” not “SCHLANGEN”). Do not format title and section headings in all capitals as well except for proper names (such as “BLEU”) that are conventionally in all capitals. The affiliation should contain the author’s complete address, and if possible, an electronic mail address. Start the body of the first page 7.5 cm from the top of the page.

The title, author names and addresses should be completely identical to those entered to the electronical paper submission website in order to maintain the consistency of author information among all publications of the conference. If they are different, the publication chairs may resolve the difference without consulting with you; so it is in your own interest to double-check that the information is consistent.

**Abstract**: Type the abstract at the beginning of the first column. The width of the abstract text should be smaller than the width of the columns for the text in the body of the paper by about 0.6 cm on each side. Center the word **Abstract** in a 12 point bold font above the body of the abstract. The abstract should be a concise summary of the general thesis and conclusions of the paper. It should be no longer than 200 words. The abstract text should be in 10 point font.

**Text**: Begin typing the main body of the text immediately after the abstract, observing the two-column format as shown in the present document. Do not include page numbers.

**Indent** when starting a new paragraph. Use 11 points for text and subsection headings, 12 points for section headings and 15 points for the title.

### 7.6 Sections

**Headings**: Type and label section and subsection headings in the style shown on the present document. Use numbered sections (Arabic numerals) in order to facilitate cross references. Number subsections with the section number and the subsection number separated by a dot, in Arabic numerals. Do not number subsubsections.

**Citations**: Citations within the text appear in parentheses as (**?**) or, if the author’s name appears in the text itself, as Gusfield (**?**). Append lowercase letters to the year in cases of ambiguity. Treat double authors as in (**?**), but write as in (**?**) when more than two authors are involved. Collapse multiple citations as in (**?**; **?**). Also refrain from using full citations as sentence constituents. We suggest that instead of

> “(**?**) showed that ...”

you use

> “Gusfield (**?**) showed that ...”

If you are using the provided LaTeX and BibTeX style files, you can use the command `\newcite` to get “author (year)” citations.

As reviewing will be double-blind, the submitted version of the papers should not include the authors’ names and affiliations. Furthermore, self-references that reveal the author’s identity, e.g.,

> “We previously showed (**?**) ...”

should be avoided. Instead, use citations such as

> “Gusfield (**?**) previously showed ... ”

**Please do not use anonymous citations** and do not include acknowledgements when submitting your papers. Papers that do not conform to these requirements may be rejected without review.

**References**: Gather the full set of references together under the heading **References**; place the section before any Appendices, unless they contain references. Arrange the references alphabetically by first author, rather than by order of occurrence in the text. Provide as complete a citation as possible, using a consistent format, such as the one for *Computational Linguistics* or the one in the *Publication Manual of the American Psychological Association* (**?**). Use of full names for authors rather than initials is preferred. A list of abbreviations for common computer science journals can be found in the ACM *Computing Reviews* (**?**).

The LaTeX and BibTeX style files provided roughly fit the American Psychological Association format, allowing regular citations, short citations and multiple citations as described above.

**Appendices**: Appendices, if any, directly follow the text and the references (but see above). Letter them in sequence and provide an informative title: **Appendix A. Title of Appendix**.

### 7.7 Footnotes

**Footnotes**: Put footnotes at the bottom of the page and use 9 points text. They may be numbered or referred to by asterisks or other symbols.[1] Footnotes should be separated from the text by a line.[2]

### 7.8 Graphics

**Illustrations**: Place figures, tables, and photographs in the paper near where they are first discussed, rather than at the end, if possible. Wide illustrations may run across both columns. Color illustrations are discouraged, unless you have verified that they will be understandable when printed in black ink.

**Captions**: Provide a caption for every illustration; number each one sequentially in the form: "Figure 1. Caption of the Figure." "Table 1. Caption of the Table." Type the captions of the figures and tables below the body, using 11 point text.

## 8 XML conversion and supported LaTeX packages

Following ACL 2014 we will also we will attempt to automatically convert your LaTeX source files to publish papers in machine-readable XML with semantic markup in the ACL Anthology, in addition to the traditional PDF format. This will allow us to create, over the next few years, a growing corpus of scientific text for our own future research, and picks up on recent initiatives on converting ACL papers from earlier years to XML.

We encourage you to submit a ZIP file of your LaTeX sources along with the camera-ready version of your paper. We will then convert them to XML automatically, using the LaTeXML tool (`http://dlmf.nist.gov/LaTeXML`). LaTeXML has *bindings* for a number of LaTeX packages, including the ACL 2015 stylefile. These bindings allow LaTeXML to render the commands from these packages correctly in XML. For best results, we encourage you to use the packages that are officially supported by LaTeXML, listed at `http://dlmf.nist.gov/LaTeXML/manual/included.bindings`

## 9 Translation of non-English Terms

It is also advised to supplement non-English characters and terms with appropriate transliterations and/or translations since not all readers understand all such characters and terms. Inline transliteration or translation can be represented in the order of: original-form transliteration "translation".

[1]This is how a footnote should appear.

[2]Note the line separating the footnotes from the text.

## 10 Length of Submission

Long papers may consist of up to 8 pages of content, plus two extra pages for references. Short papers may consist of up to 4 pages of content, plus two extra pages for references. Papers that do not conform to the specified length and formatting requirements may be rejected without review.

## Acknowledgments

The acknowledgments should go immediately before the references. Do not number the acknowledgments section. Do not include this section when submitting your paper for review.

## References

### 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 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.