Applied NLP & Machine Learning: Video Game Reviews

Matt Mulholland

Background

- Deception Detection class (Dr. Fitzpatrick)
  - Application: detecting fake reviews (for Amazon.com, hotel websites, etc.)
- Idea: Apply same ideas to video games
  - Steam
    - Online video game platform widely used to play, buy, and review games and that functions also as a social media platform
    - One intriguing piece of data: number of hours played, i.e., experience
    - Also: number of times each review was marked as helpful, funny, number of friends the reviewer has, number of reviews he/she has written, etc.
  - Detecting fake reviews --> relating reviews to experience
    - Related, but not the same
    - If user has little experience with a game, that should show up in their review (forget whether or not the review is fake)

Problem

Question

Can the relationship between reviews and reviewer experience be modeled?

Aim

Generate experience predictions for reviews --> ranking/filtering

Application

Generalize models for when information about experience is lacking (either because it can't be or wasn't collected)

Code

Stored in an MIT-licensed GitHub repository

- Web scraping + NLP/machine learning pipeline for conducting iterative machine learning experiments
- Python/Cython (C)
- Almost 600 commits (changes/additions/updates) to the codebase
- 5,400+ lines of Python/Cython
- Two other contributors (Janette Martinez & Emily Olshefski)
- Jupyter (formerly known as IPython) notebooks explaining some of the thought process behind the code
- Easy to use (if you have access to a computer with 40+ GB of RAM and an HDD that can devote 200+ GB to a database)

Getting the Data from Steam

- Jupyter notebook
- Scraping code/notebook
- Scraping algorithm used to collect:
  - ~80,000 reviews/hours played data (+ many other pieces of user info)
  - From 11 games of various genres
    - Arma 3, Dota 2, Football Manager 2015, Team Fortress 2, The Elder Scrolls V, Garry's Mod, Counter Strike, Counter Strike: Global Offensive, Grand Theft Auto V, Sid Meier's Civilization 5, and Warframe
- Data made publicly available in an MIT-licensed GitHub repository here (posted on `reddit`)

Feature Extraction + Storage

- Storing all data in MongoDB database
    - Filter out non-English reviews (using `langdetect` in Python)
- Created an efficient pipeline for extracting NLP features and uploading them to the DB to be used later
    - Using `spaCy` (industrial-strength NLP in Python) + Cython
- Constantly running database that is always ready for machine learning experiments or other types of research

Interacting with the DB

In [4]:
```
# Connect to reviews collection
db = connect_to_db(host='localhost', port=37017)
```

In [13]:
```
cursor = db.find({'game': 'Arma_3'}, {'nlp_features': False})
```

In [14]:
```
next(cursor)
```

Out[14]:

```
{'_id': ObjectId('560394d3cbb14611d09582f9'),
 'achievement_progress': {'num_achievements_attained': 222,
  'num_achievements_percentage': 0.42857142857142855,
  'num_achievements_possible': 518},
 'appid': '107410',
 'bin_factor': 2.0,
 'bin_ranges': [[0.0, 338.1], [338.2, 1014.4], [1014.5, 2367.0]],
 'binarized': True,
 'date_posted': 'Feb 1, 2015, 3:40PM',
 'date_updated': None,
 'found_helpful_percentage': 0.5,
 'friend_player_level': 5,
 'game': 'Arma_3',
 'id_string': '560394d3cbb14611d09582f9',
 'nbins': 3,
 'num_badges': 4,
 'num_comments': 0,
 'num_found_funny': 0,
 'num_found_helpful': 1,
 'num_found_unhelpful': 1,
 'num_friends': 13,
 'num_games_owned': 49,
 'num_groups': 2,
 'num_guides': 0,
 'num_reviews': 4,
 'num_screenshots': 1,
 'num_voted_helpfulness': 2,
 'num_workshop_items': 0,
 'orig_url': 'http://steamcommunity.com/app/107410/homecontent/?userreviewsoffset=5760&p=1&itemspage=577&screenshot
spage=577&videospage=577&artpage=577&allguidepage=577&webguidepage=577&integratedguidepage=577&discussionspage=57
7&appid=107410&appHubSubSection=10&appHubSubSection=10&l=english&browsefilter=toprated&filterLanguage=default&searc
hText=&forceanon=1',
 'partition': 'training',
 'profile_url': 'http://steamcommunity.com/id/-A02-',
 'rating': 'Recommended',
 'review': 'great game, massive improvement over arma 2, the editor mode is great because you can set up your own m
issions or just blow stuff up if you want. also there is a great modding communtity',
 'review_url': 'http://steamcommunity.com/id/-A02-/recommended/107410/',
 'steam_id_number': '-A02-',
 'total_game_hours': 316.5,
 'total_game_hours_bin': 1,
 'total_game_hours_last_two_weeks': 6.7,
 'username': 'AustraliumOxide'}
```


# of users who found the review helpful

review

hours played

Features

NLP Feature Types

- word *n*-grams (for $n = 1$ to 2)
- character *n*-grams (for $n = 2$ to 5)
- syntactic dependency relationships
- length (log_2(total # of characters in review))
- Brown corpus cluster IDs

Features: *n*-grams

Word *n*-grams

Lower-casing, fixing some common unpunctuated contractions ("im" --> "i am"), tokenization, *n*-gram extraction

Example

- **Input:** "This is a great game."
- **Lower-casing + tokenization:**
  - 1-grams: ["this", "is", "a", "great", "game", "."]
  - 2-grams: ["this is", "is a", "a great", "great game", "game ."]

Features: *n*-grams

Character *n*-grams

- Case left unchanged, no preprocessing (allows for emoticons, etc.)
- Effectively deals with misspellings --> common subsequences of characters

Example

- **Input:** "I love it."
    - 2-grams: ["I ", " l", "lo", "ov", "ve", "e ", " i", "it", "t."]
    - 3-grams: ["I l", " lo", "lov", "ove", "ve ", "e i", " it", "it."]
    - 4-grams: ["I lo", " lov", "love", "ove ", "ve i", "e it", " it."]
    - 5-grams: ["I lov", " love", "love ", "ove i", "ve it", "e it."]

Features: syntactic dependencies

Preprocessing, sentence and word tokenization, POS-tagging, parsing

Examples

- **Input:** "Warframe is a great game."
  - "great" modifies "game" (could also be in "This game is great")
  - "great game" syntactically related to "Warframe"
- **Input:** "I hate Warframe."
  - "Warframe" is the direct object of "hate"
    - Could also be extracted in constructions like "Warframe is universally hated."

Features: Brown corpus cluster IDs

- Preprocessing, word tokenization
- Each content word is related to a specific ID associated with a cluster of the Brown corpus
- Broadly speaking, this could contribute to the review modelling by profiling each review in terms of the vocabulary distribution

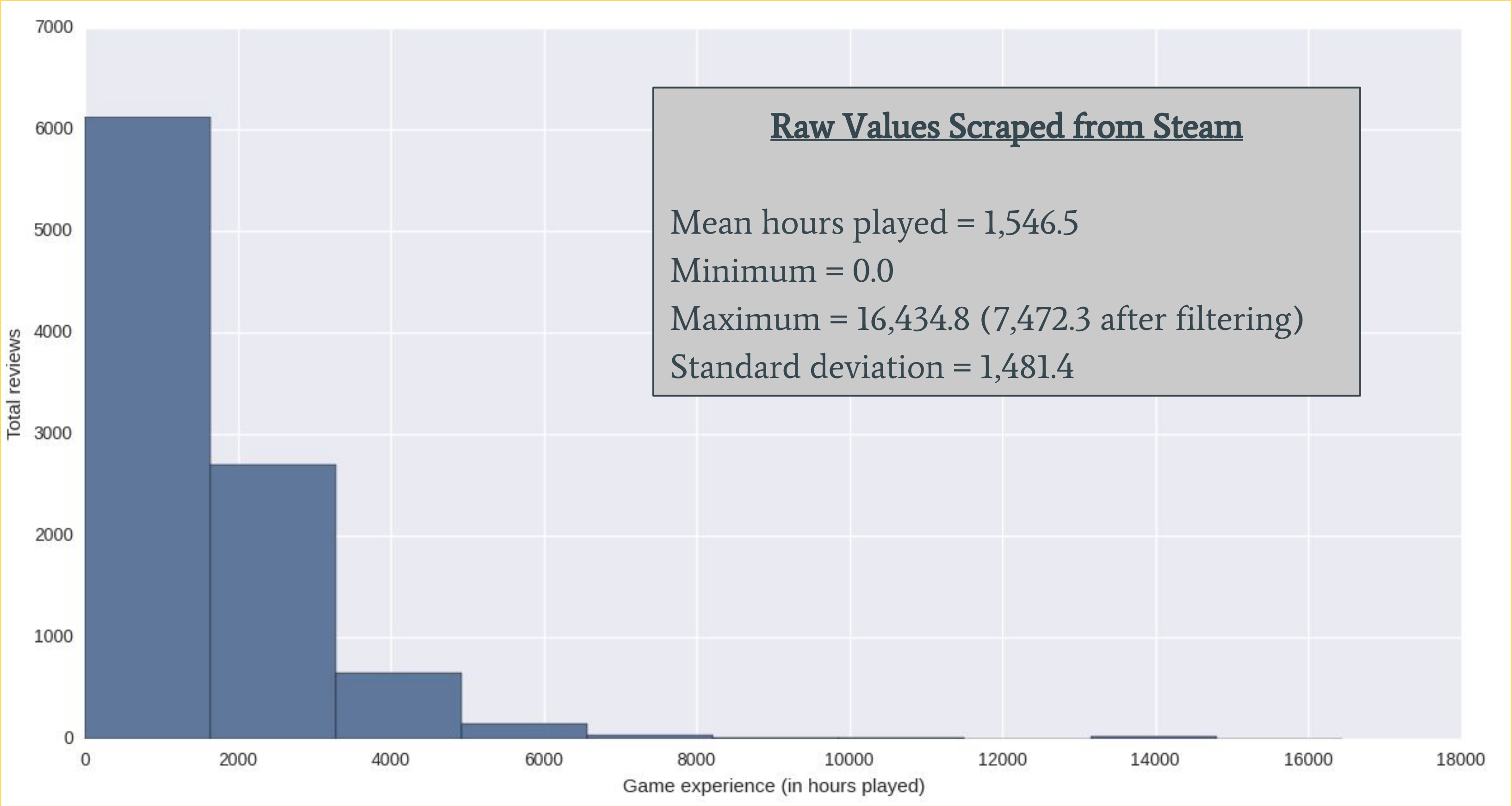
`learn.py`

- Main utility for conducting machine learning experiments
- Iterative learning with "grid search"
- Configuring experiments:
  - Use multiple `scikit-learn` learning algorithms
    - `Perceptron`, `PassiveAgressiveRegressor`, `MiniBatchKMeans`, `BernoulliNB`, and `MultinomialNB`
  - Learn relationship between the review and hours played or ANY other attribute (friends, times marked helpful, etc.)
  - Collapse raw label values into "bins" of varying or equal sizes
  - Manipulate the label distribution with *ln* or exponential functions
  - Use NLP features and/or attribute values to train models
  - Train/test with different sets of games simultaneously

Experiments

- Raw, unscaled label values
- Total game hours played
  - Highly positively skewed distribution (characteristic of most of the labels, in fact)
  - Most of data is contained in a small portion of the histogram
  - So, filtering was applied before the data was used in actual experiments

Dota 2: Game Hours Distribution



Experiments

- Even after filtering out outliers, the distribution of raw label values is still problematic
  - To perform the usual types of analyses (kappa, confusion matrices, etc.), values need clustering
  - Perhaps need even more normalization and/or sophisticated statistical analysis
  - Solution: Implement an algorithm for clustering the values into "bins" and allow a "factor" to be set such that the size of the bins can increase or decrease as the values increase (or the bins can be evenly-sized)

Experiments

- Primary purpose: Study the relationship between how many hours a reviewer played a game and the review text (and attributes of the review/reviewer)
- Let's take a look at an example experiment in detail to see how everything works in simplified form: Running an incremental learning experiment with `util.learn.RunExperiments`

Experiments: Game Hours

- Raw label values collapsed into 3 bins with a factor of 2.0
  - The 2nd bin is twice as large as the first, the 3rd twice as large as the second
  - However, this does not completely rectify the distribution issues: the distribution of binned values is still highly positively skewed
- Think of it as:
  - 1 = low experience
  - 2 = average experience
  - 3 = high experience

Experiments: Game Hours Summary

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	0.60	0.57	-0.01	-0.06
Arma 3	non-NLP	Perceptron	0.52	0.56	-0.02	-0.08
CS	majority label		0.81	0.66	n/a	n/a
CS	NLP+non-NLP	MiniBatchKMeans	0.60	0.71	0.16	0.06
CS	NLP	MultinomialNB	0.82	0.69	0.16	0.28
Dota 2	majority label		0.50	0.25	n/a	n/a
Dota 2	NLP+non-NLP	Perceptron	0.53	0.49	0.08	-0.15
Dota 2	non-NLP	PassiveAgressive	0.42	0.47	0.16	0.31

Experiments: # Times Marked "Helpful"

- Train a system to predict how "helpful" a review is
- Raw label values collapsed into 3 bins with a factor of 6.0
  - 2nd bin 6 times larger than 1st and so forth...

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	0.00	0.00

Future

- Do a lot of experimental rounds with different labels
- Integrate a more sophisticated statistical approach:
    - Transforming the range of raw label values into a set or scale of discrete labels
- Pull out the more general pieces of the code:
    - Instead of a machine learning + NLP + Steam reviews extraction system, make a general iterative machine learning system that can be used with any kind of data
- Separate out the NLP component and make it extensible
- Design NLP features that come closer to modeling the aspects of good reviews demonstrating experience

Contact

- Matt Mulholland (mulhodm@gmail.com)
- Please feel free to contact me with any questions! Thanks.