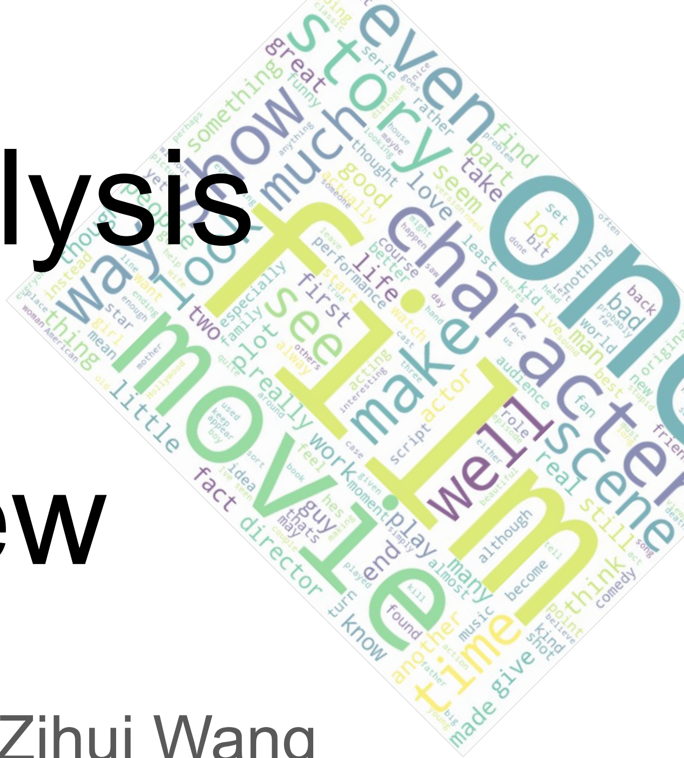
Sentiment Analysis on Movie Review



Shihong Wei, Jingyi Ren, Xuan Wu, Zihui Wang
swei15, jren20, xwu78, zwang220

Track: application
Sub-track: Texture Analysis

Problem Definition

Problem Formulation: Predict whether a movie review is positive or negative.

- Identify key information in reviews that influences (or reflects) sentiments
- Evaluate Fairness of Models

Data Types in the Problem:

- Instance input: texture movie reviews
- Instance label: positive or negative

Connection with the course:

- Supervised Learning (Binary Classification Problem)
- Feature Engineering & Dimension Reduction: Vectorization of texture data, PCA
- Methods: Logistic Regression, SVM, Neural Networks

Motivation

Real-World Implication:

- Provide reference for the audience when making movie-watching decisions
- Help the production company to analyze the market situation

Similarity to Lecture/Breakout/Homework:

- HW 3 Lab (Classification)/ HW4 Program (Kernel SVM on Texture Data)/ Texture Analysis

What Makes this Problem Unique?

- Selection of newly-released movies
- Summarize comments from various movie genres, runtimes, websites

Ethical Implications

- Representative of English speakers, not opinions from viewers of other cultures
- Public relations or paid posters may write fake reviews
- Possible negative (positive) social impact on movie castings

Dataset Description

Main: Large Movie Review Dataset

- A collection of English textual movie reviews extracted before year 2011 from IMDB
- Size of dataset: 35000 training; 7500 validation; 7500 testing

Extra Testing Sets:

- Extract reviews for movies of different genres and runtime on IMDB website for movies since 2013 using web crawler algorithm
- Size: 500 reviews for each of 8 genres and runtime (<100 minutes; >100 minutes)

Review Instances:

- Positive Instance: If you like original gut wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it.

Great Camp!!!
- Negative Instance: A complete waste of time. Nonsense and silliness from the beginning till the end.

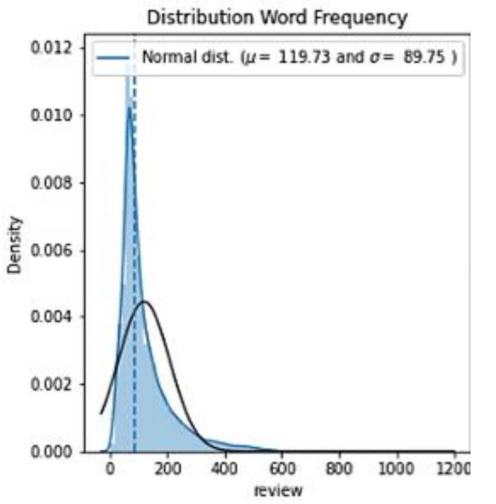
Pre-Processing and Feature Engineering

Pre-Processing:

- Review text:
  - removing html strips, square brackets, noisy text, special characters, stopwords;
- Label: coding the labels into 0,1

Feature Engineering:

- Vectorization:
  - Bag of word (2-word, 3-word…): `sklearn.feature_extra`
  - NGram of Characters: `sklearn.feature_extraction`
  - TFIDF: `sklearn.feature_extraction`
  - Word2vec: `gensim.models`
- Dimension Reduction:
  - PCA: `sklearn.decomposition`



Methods

Main Methods:

- Baseline Method: Logistic Regression(LR) + Bag-of-Words
- Other Methods: LR, SVM, CNN and LSTM with different vectorization techniques (word2vec, Character N-Gram, TFIDF), and dimension reduction (PCA)

Why Choose These Methods?

- Textual input data: Different Vectorizations
- Large Dataset: Dimension reduction including PCA
- ‘Positive’ or ‘Negative’ Label: Binary classification problem (LR, SVM, CNN, LTSM)

How to Train These Methods?

- Model libraries: `sklearn.linear_model`: LR
  `sklearn.svm`: SVM, `keras`: CNN and LSTM
- Model selection within each class: hyperparameter grid search `sklearn.model_selection`

Methods

Evaluation of Models

- Loss function: hinge, squared hinge with regularization (l1, l2)
- Performance Measure: TN, FP, FN, TP, Accuracy, Precision, Recall, Specificity, FPR, F1 Score, Balanced Accuracy
- Other criterion: time complexity, ease of implementation, fairness and interpretability

Difficulty: LR and SVM (relatively easier) CNN (best architecture), LSTM (time-consuming)

Established Baselines in Previous Studies:

- Term Frequency Vectorization+CNN `keras.datasets.imdb`
  - `https://github.com/ovguyo/moviereview`
- {1,2}-Bag of Words Vectorization+Random Forest `keras.datasets.imdb`
  - `https://www.kaggle.com/ramanchandra/sentiment-analysis-on-imdb-movie-reviews/data`

Results

Comparison with the Baseline:

Methods	word2vec +SVM	word2vec +LR	NGram +SVM	NGram +LR	TFIDF+ SVM	TFIDF+ LR	Baseline (Bag of Word)+LR
Accuracy on Test Split	0.86	0.854	0.79	0.84	0.86	0.859	0.82, 2-word bag 0.69, 3-word bag

- Results for NN models to be updated

Surprising Results:

- word2vec is not superior when compared with other vectorizations
- Performance of LR are quite good even if it a relatively simpler method
- potential reason:
    - Hyperparameter grid is not large enough during the grid search
    - Maybe the models have overfitting problems

Deliverables

What we have done on deliverables:

- Appropriate data preprocessing and vectorization (bag of word, N-Gram, TFIDF, and word2vec).
- Ensure fair comparison and to avoid information leakage.(training-validation-testing splits)
- Exploration of model interpretation in textual analysis (e.g. Identify keywords from TFIDF score)
- Grid search and tuning hyperparameters (for logistic and SVM)
- High test accuracy of models for logistic and SVM
- Model comparison across different classes for logistic and SVM based on various criterion
- Try out Web crawler algos to obtain extra testing sets from IMDB to test group difference
- Further evaluation of the model performance and fairness across different groups

One Test Output Example: TFIDF + PCA + LR

Extra testing sets extracted from IMDB after year 2013

Criterion	Test_Split	Runtime_1_100	Runtime_101_600	Action	Adventure	Animation	Biography	Comedy	Horror	Romance	Sci_fi
TN	3226.000	164.000	148.000	173.000	165.000	61.000	110.000	158.000	205.000	136.000	186.000
FP	570.000	86.000	102.000	77.000	85.000	55.000	121.000	92.000	45.000	114.000	64.000
FN	487.000	66.000	41.000	65.000	47.000	10.000	27.000	39.000	72.000	40.000	81.000
TP	3217.000	184.000	209.000	185.000	203.000	106.000	204.000	211.000	178.000	210.000	169.000
Accuracy	0.859	0.696	0.714	0.716	0.736	0.720	0.680	0.738	0.766	0.692	0.710
Precision	0.849	0.681	0.672	0.706	0.705	0.658	0.628	0.696	0.798	0.648	0.725
Recall	0.869	0.736	0.836	0.740	0.812	0.914	0.883	0.844	0.712	0.840	0.676
Specificity	0.850	0.656	0.592	0.692	0.660	0.526	0.476	0.632	0.820	0.544	0.744
FPR	0.150	0.344	0.408	0.308	0.340	0.474	0.524	0.368	0.180	0.456	0.256
F1	0.859	0.707	0.745	0.723	0.755	0.765	0.734	0.763	0.753	0.732	0.700
Balanced_Accuracy	0.849	0.669	0.632	0.699	0.682	0.592	0.552	0.664	0.809	0.596	0.734

One Test Output Example: Word2Vec + LR

Criterion	Test_Split	Runtime_1_100	Runtime_101_600	Action	Adventure	Animation	Biography	Comedy	Horror	Romance	Sci_fi
TN	3218.000000	164.000000	151.000000	176.000000	165.000000	55.000000	102.000000	152.000000	202.000000	127.000000	196.000000
FP	578.000000	86.000000	99.000000	74.000000	85.000000	61.000000	129.000000	98.000000	48.000000	123.000000	54.000000
FN	518.000000	69.000000	39.000000	64.000000	55.000000	13.000000	31.000000	46.000000	70.000000	38.000000	87.000000
TP	3186.000000	181.000000	211.000000	186.000000	195.000000	103.000000	200.000000	204.000000	180.000000	212.000000	163.000000
Accuracy	0.853867	0.690000	0.724000	0.724000	0.720000	0.681034	0.653680	0.712000	0.764000	0.678000	0.718000
Precision	0.846440	0.677903	0.680645	0.715385	0.696429	0.628049	0.607903	0.675497	0.789474	0.632836	0.751152
Recall	0.860151	0.724000	0.844000	0.744000	0.780000	0.887931	0.865801	0.816000	0.720000	0.848000	0.652000
Specificity	0.847734	0.656000	0.604000	0.704000	0.660000	0.474138	0.441558	0.608000	0.808000	0.508000	0.784000
FPR	0.152266	0.344000	0.396000	0.296000	0.340000	0.525862	0.558442	0.392000	0.192000	0.492000	0.216000
F1	0.853240	0.700193	0.753571	0.729412	0.735849	0.735714	0.714286	0.739130	0.753138	0.724786	0.698073
Balanced_Accuracy	0.853943	0.690000	0.724000	0.724000	0.720000	0.681034	0.653680	0.712000	0.764000	0.678000	0.718000

Deliverables

Difficulties:

- Finding good architecture for CNN and LSTM with reasonable time complexity
- Model interpretation for N-Gram since it is character-wise vectorization

Adjustment on deliverables:

- Edit one deliverable: when implement NGram, TFIDF and models, we use package `keras`, `sklearn.feature_extraction`
- Delete “avoid redundant intermediate computation” when writing iterations

Reason:

- Accelerate model implementation and focus more on analysis, rather than spending too much time on repetition of previous programming hws
- For ease of grid search of hyperparameters (more user-specified hyperparameters)
- Fair comparison

What We’ve Learned

Concepts from lecture/breakout most relevant to the project:

- Supervised Learning (Binary Classification)
- Valid Modeling Procedure: Train-Valid-Test Split….
- Feature Engineering: Vectorization of texture data (Word of bags, Ngram, TFIDF)
- Dimension Reduction: PCA
- Methods: Logistic Regression, SVM, CNN, LSTM
- Model interpretation and Model fairness evaluation

The aspects of our project that are most surprising:

- The fact that we can use ML algo to predict sentiments of movie reviews with quite high accuracy even with the diverse difference in people’s expressions
- Even relatively simple model can do well than complex methods such as NN under some circumstances

What We've Learned

What we would do differently if you were going to start from the beginning:

- Using web crawlers to extract a more recent training set from different websites
  - The IMDB large movie review dataset is constructed at 2011

Questions we still have:

- Any classic model interpretation techniques for texture data (e.g., for image data, we have learn the LIME), so that we can try to explain why certain instance is misclassified?
- How to construct a NN architecture with high test accuracy with different vectorizations?

What would be the most helpful feedback to get from other groups:

- Reasoning Behind Group Fairness Difference
- Suggestion on construct a good NN architecture

References

Papers

1. Maas, A.L. & Daly, R.E. & Pham, P.T. & Huang,D. & Ng, A.Y & Potts,C (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011). Available at https://ai.stanford.edu/ amaas/data/sentiment/
2. Mikolov, T. & Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
3. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Software Packages (Main Python Libraries)

- `numpy, pandas, sklearn, gensim, keras, nltk, bs4, seaborn, matplotlib, textblob, wordcloud`
- For python webclawer algorithm on IMDB, please see
  - `https://shravan-kuchkula.github.io/scrape_imdb_movie_reviews/#`