Laughter Through the Decades: Replicating Humor Detection Research

Joshua Hoeflich
Northwestern University / 1881 Oak Avenue, Evanston IL
joshuahoeflich2021@u.northwestern.edu

Abstract

We attempt to replicate "Making Computers Laugh: Investigations in Automatic Humor Recognition" by applying the techniques within it for classifying humorous and non-humorous texts to publicly available data sets. [6] We compare the performance of three models: A naive Bayes classifier, an SVM classifier, and a decision tree classifier that uses heuristics based on the presence of antonyms, adult slang, and sounds such as rhymes and alliteration within documents. We train and compare these models on a collection of short jokes, Jeopardy questions, news headlines, and Amazon food reviews. We find that despite substantive differences in the datasets and implementations in our work and the original paper, our systems perform comparably well at the same tasks.

1 Introduction

Few other tasks illustrate the astonishing effectiveness of machine learning as humor recognition. Comedy seems so subjective and points to so many unanswered deep philosophical questions that, at first glance, the notion of teaching a machine to recognize jokes appears absurd. However, in 2005, computational linguists Rada Mihalcea and Carlo Strapparava published a paper titled "Making Computers Laugh: Investigations in Automatic Humor Recognition" showing that Naive Bayes and Support Vector Machine (SVM) classifiers could do a remarkably good job at distingushing one-liners from many other kinds of texts. The strength of their results merits deeper examination.

In this article, we describe an attempt to replicate the results of "Making Computers Laugh" over 15 years after its original publication. We describe the steps involved with mimicking their processes and the challenges we faced while trying to do so, taking special note of the places using the methods described in the original paper proved infeasible. We then evaluate our new results in comparison with the original ones and conclude with a discussion of relevant related works.

2 Data

Unfortunately, as of 2021, the exact data sets used in "Making Computers Laugh" are unavailable, which makes replicating the paper somewhat challenging. The article describes an elaborate web scraping process used to collect 16000 one-liners, but the links it uses to describe that method no longer point to live websites. The paper also does not link to the "online proverb collection" it uses for examples of non-humorous input data; it does not describe precisely which Reuters headlines or British National Corpus sentences it uses.

Accordingly, our first task involved finding data different but comparable to that used in the article. For positive examples of humor, we used a collection of over 200,000 short jokes; for negative examples, we used a list of 200,000 Jeopardy questions, a list of 1,000,000 news headlines from the Australian Broadcasting Corporation (ABC), and 500,000 reviews taken from Amazon.com about fine foods. [8] [5] [13] [11] All of these datasets came from Kaggle, a website with numerous other extensive datasets and machine learning models. While these datasets are not precisely analogous with the ones used in the original paper, they present a comparably diverse set of text with which to explore the techniques from the article.

Using short jokes instead of one-liners as in the original paper is appropriate because both forms of data work well with classifiers for the same reason. The original authors note: "While longer jokes can have a relatively complex narrative structure, a one-liner must produce the humorous effect 'in one shot,' with very few words. These characteristics make this type of humor particularly suitable for

use in an automatic learning setting" (Mihalcea and Strapparava, 2005). The short jokes selected for this replication are not necessarily guaranteed to be exactly one sentence long. Still, their similar length means that our system will likely perform as well or as badly on them for reasons analogous to those explored in the original paper.

3 Methods

With several datasets at hand, the next step of replicating the paper required creating and training several different machine learning models. We started by creating a decision tree classifier that relied on embedding documents into vectors with numeric components derived from up to three heuristics: the count of antonyms, the count of words related to sexuality, and the count of alliterations and rhymes found in a given document. No off-the-shelf mapping between those heuristics and vectors existed, so we needed to implement our own based on the content of the paper.

Finding antonyms involved using the semantic information stored in Princeton's WordNet lexical database. [7] The paper used this dataset as well, though they may have relied on an older version; we used the one directly available through version 3.6.5 of the Natural Language Toolkit (NLTK) Python library. [1] We also used the version of Carnegie Mellon's pronunciation dictionary bundled with NLTK to count the number of rhymes and alliterations within a sentence. [1] Implementing these counts required writing a few functions that tokenized the sentences, stripped out their stopwords, and tracked the relevant features of each heuristic using the APIs that NLTK exposes directly.

Tracking the words related to sexuality proved more challenging. "Making Computers Laugh" relied on traversing a graph constructed with help from the WordNet Domains project, which associated items from WordNet with relevant domain labels. More specifically, the authors performed a graph search by finding words related to one another through the SEXUALITY label in that project. Unfortunately, the link to download WordNet Domains no longer appears to work, and while an unofficial copy of the project exists on GitHub, the data within it appears out of date with the most modern WordNet corpus. [2] Because of these challenges, we opted to find words related to sexuality differently: We relied on pre-computed GloVe vectors for word representation. [9] We wrote a script to find the top 5,000 words with the closest cosine similarity to the word SEXUALITY and counted the number of times they appeared within a document. As we shall see in the results section, this strategy appears to have yielded analogous results to those in the original paper.

After implementing these heuristics, we had the information we needed to begin running the three machine learning models. The scikit-learn library provided both high-quality off-the-shelf tools for transforming raw documents into TFIDF vectors and using those vectors to learn with Naive Bayes and an SVM classifiers. [2] Replicating the paper in those circumstances simply involved running those tools.

4 Results

We now present our results using the decision tree classifier with vectors extracted from the heuristics. We trained the model using 1000 samples and tested it on 15000 taken from each of the three datasets; the mean accuracy on the test sets are below.

Heuristic	Jokes/ Reviews	Jokes/ Jeopardy	Jokes/ Headlines
Adult Slang	71.32%	60.32%	69.06%
Sounds	65.23%	52.67%	57.77%
Antonyms	51.92%	51.31%	52.07%
All	73.82%	60.16%	70.21%

TABLE 1: Humor recognition mean accuracies using adult slang words, sounds (such as rhymes and alliteration), antonyms, and all heuristics at once.

This range of percentages (70% to 50%) roughly matches the data in the original paper.

We now describe our results using Naive Bayes and SVM classifiers. In these examples, we also used 1000 training samples and 15000 test samples. The results were as follows:

Technique	Jokes/ Reviews	Jokes/ Jeopardy	Jokes/ Headlines
Naive Bayes	90.82%	88.81%	94.35%
SVM	93.37%	92.99%	97.13%

[1] See CMU dict's official website.
[2] See WordNet Domain's website and this repository.

TABLE 2: Humor recognition mean accuracies using Naive Bayes and SVM classifiers.

These scores are comparable to the ones in the original paper, albeit somewhat higher in the worst-case scenarios.

5 Discussion

Intriguingly, in the heuristics part of this experiment, we found that words related to sexuality served as the most effective indicator of humorous content, whereas in the original paper alliteration and rhymes yielded the most accurate clues. Differences in the jokes dataset may have caused this discrepancy; jokes explicitly labeled as one-liners are probably more likely to rely on sound-based humor than those that are specifically short. Furthermore, our decision to use GloVe vectors instead of manually labeled documents may have contributed to the effectiveness of that measure as well, as the word embeddings may have captured additional information missing in the domain labels the original paper used.

Jeopardy questions likely presented the greatest challenge to all of the models because of their semantic similarity to jokes. While news headlines and food reviews present rather straightforward factual information, Jeopardy questions necessarily point to a punchline in the guise of a question that a contestant must create. Jeopardy questions also seem more likely to use unusual language and words that would appear in jokes but not in the other two contexts. As such, the drop in performance seems intuitive.

Given the numerous differences in implementation within this project and the original text, that these content-based classifiers achieved equally good or even better results suggests that the techniques and findings used in "Making Computers Laugh" do indeed replicate: classic classification techniques can in some circumstances successfully distinguish between short humorous and non-humorous documents from various corpora. While the numbers changed, the general patterns stayed the same, which indicates that the ideas of the paper merit further exploration.

6 Related Work

Other researchers have cited "Making Computers Laugh" in further work in humor recognition. For example, in "That's What She Said: Double Entendre Identification", the authors used the paper as inspiration for a system that could detect whether saying "That's what she said" would make a sentence funny or not (Kiddon and Brun, 2011) [4]. In "Humor: Prosody Analysis and Automatic Recognition for F * R * I * E * N * D * S *", the authors used the paper as inspiration for a system that applied classifiers to the scripts of the T.V. show *Friends* annotated with laugh tracks to detect which lines were and were not funny (Purandare and Litman, 2006) [12].

Computational linguists working on problems other than humor recognition have found "Making Computers Laugh" helpful as well. For example, in "Semi-Supervised Recognition of Sarcastic Sentences in Twitter and Amazon", the authors took inspiration from the Naive Bayes and SVM classifiers when trying to implement a system that could detect sarcasm in documents (Davidov, Tsur, and Rapport, 2010) [3]. Moreover, in "Automatic Detection of Fake News", the authors cite the paper when describing the challenges outlets like *The Onion* pose when trying to detect nonfactual news stories (Pérez-Rosas et. al, 2017) [10]. That "Making Computers Laugh" had an impact on articles grappling with the enormous influence of social media illustrates the paper's importance. By helping others understand some of the most contentious and important language of our time, "Making Computers Laugh" shows that humor recognition is no laughing matter.

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition, 2009.

[2] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[3] D Davidov, O Tsur, and A Rappoport. Semi-supervised recognition of sarcastic sentences in twitter and amazon (pp. 15–16). *Retrieved from Association for Computational Linguistics website: https://www. aclweb. org/anthology/W10-2914. pdf*, 2010.

[4] Chloe Kiddon and Yuriy Brun. That's what she said: double entendre identification. In *Proceedings of the*

49th annual meeting of the association for computational linguistics: Human language technologies, pages 89–94, 2011.

[5] Rohit Kulkarni. A million news headlines, January 2021.

[6] Rada Mihalcea and Carlo Strapparava. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538, 2005.

[7] George A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, nov 1995.

[8] Abhinav Moudgil. Short jokes, March 2017.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[10] Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*, 2017.

[11] Stanford Network Analysis Project. Amazon fine food reviews, May 2017.

[12] Amruta Purandare and Diane Litman. Humor: Prosody analysis and automatic recognition for f* r* i* e* n* d* s. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215, 2006.

[13] Bojan Tunguz. 200,000+ jeopardy! questions, November 2019.