COSI 140 Collaborative Project - Final Report

Joseph Antaki, Eli Goldner, Angus L'Herrou

May 13, 2020

Summary

This project is based on the *#HashtagWars* corpus from Task 6 of SemEval-2017. The corpus consists of tweets posted for viewer competitions in the game show *@midnight*. The original corpus is annotated for whether a tweet was in the top ten funniest tweets or the winning tweet of a given hashtag. We worked to assign a deeper classification of humor to these tweets, toward the goal of identifying specific types of humor in a tweet.

Dataset Description

Our gold standard dataset, compiled from three different annotators, working in pairs on a given tweet hashtag, and checked for quality and accuracy of annotation, consists of 1738 total tweets from 14 different tweet categories. Each tweet is annotated as falling into one or more of the following humor categories:

1. **Absurdism**
2. **Insult**
3. **Irony**
4. **Observational**
5. **Wordplay**
6. **Vulgarity**
7. **Other**
8. **Not Humor**

Where **Not Humor** was used if a tweet made no attempt at Humor. These categories are adapted from Hay (1995), given their use in humor categorization of tweets in Raz (2012).

Dataset Quality

We computed Inter-Annotator-Agreement using Cohen's kappa between each pair of annotators. Since the annotation task was multiclass and multilabel, we considered the power set of our classes as the agreement space. We found the following Cohen's kappa scores:

$$\kappa(A, B) = 0.32405575580187357$$

$$\kappa(B, C) = -0.021085787381538594$$

$$\kappa(A, C) = 0.12522621262712513$$

These low kappa scores along with their high variation suggest that hhumor can be subjective, with different annotators perceiving different types of humor present in the same joke. This also suggests potential for improvement in annotation specification, towards reduce subjectivity.

Machine Learning Models

We used multiple machine learning moodels to further explore which was best suited to the task, namely `sklearn`'s `MultiOutputClassifer` which transforms an instance of a `LogisticRegression` classifer to handle multiple labels, along with `sklearn`'s `MLPClassifier`. The first model being more familiar from coursework, modified to handle the possibility of multiple labels, and the second model being based on neural networks and also suited to handle the possibility of multiple labels.

Features

For detection of different categories of humor we employed different sets of features, in particular with attention to both a more generalized approach to features and the model along an appraoch more tailored to wordplay.

Generalized Approach

Feature Engineering

As a preprocessing step, the following textual elements were removed from each tweet: URL artifacts indicating some sort of media attachment in the original tweet (e.g. "https://t.co/abcxyz"), the topic hashtag (e.g. "#420Celebs"), and any mention of "@midnight", which, like the topic hashtag, was a common inclusion despite usually not being part of any joke. The removal of these irrelevant elements increased performance slightly. Following in the footsteps of Mahajan and Zaveri (2017),

a set of general features was experimented with as one alternate avenue for humor type detection. These features generally analyzed the textual form and writing style of the tweet, as well as some other information like possible double meanings and intensity of words used. The length of the

tweet in characters, the word count, and the average word length were examined. Additionally, we included occurrence counts for each of: nouns, adjectives, verbs, adverbs, and punctuation, as well as each of those parts of speech's frequencies with respect to the tweet's word count (except punctuation). Lastly, we looked at intensity scores for adjectives and adverbs. For each of those two parts of speech, the following were features: the sum of all intensity scores of the part of speech in the tweet, the average intensity score, the maximum score, and the difference between the maximum and average score. These features examined general trends in joke form and delivery.

Ideally, tweets that specifically used strong language, like Insults or Vulgarities, would have been detected by this group of features, while the forms of other types of humor would also have been accounted for. In the interest of time, instead of calculating adjective and adverb intensity in the style of Potts(2011), as cited in Mahajan and Zaveri (2017), words were looked up in SentiWordNet, and the greater of their positive and negative scores was used as their intensity scores (objective

score was ignored since objectiveness is not intense or polar in any emotionally significant way). If a word did not appear in SentiWordNet, it was ignored and not factored into the average intensity score. The next features we included were the average of the WordNet synset counts of all the

words in the tweet, the highest synset count of a word in the tweet, and the difference between the greatest and average synset values. Again, if a word had no synsets in WordNet, it was not factored into the average. These features were aimed at detecting wordplay such as semantic puns or double entendres by searching the documented multiple meanings of words in WordNet. Wordplay and Vulgarity especially were the target humor types of these features. One group of features that did

not make it to the final model was word frequencies in written and spoken corpora, taken from the American National Corpus frequency data. Using each of the two environments separately (written and spoken), the following measures were taken: average of frequencies of all words in the tweet, the lowest frequency measure of an individual word found in the tweet, and the difference between the highest and lowest individual frequencies. Lastly, the difference between the written and spoken averages was taken into account. These frequency features were intended to quantify

incongruity and unexpectedness in word use, which is a common theme in humorous utterances. The unpredictability of Absurdity and Observational humor's juxtaposition of normal things with funny comments were the intended targets of these features, but in practice, they lowered the performance of the model, so they were discarded. After implementing these features, we took

some steps of our own and added some new measurements: named entity recognition and profanity detection. For the former, we utilized spaCy's named entity detection abilities and counted the number of named entities per tweet. Sometimes, the names of places and celebrities would appear directly in a tweet, but more often there were intentional misspellings to create puns on the names. Despite this, the name still had the form and context of a normal named entity, allowing it to be detected. Observational humor and Wordplay (since puns were often on the names of people or books/movies) were the intended targets for this feature. Profanity detection was accomplished

with the `profanity-check` Python module (Zhou). The idea was that the best way to detect Vulgar tweets was to look for vulgar and offensive words directly, which is what this module accomplishes. It is a trained machine learning model in itself that calculates the probability that a string is offensive by way of profanity. This probability score was included as a feature for our humor recognition.

Model

For this general feature approach, we utilized `sklearn`'s Logistic Regression model. Since this model cannot handle multilabel output out of the box, it was supplemented with `sklearn`'s `MultiOutputClassifier`. This fits one classifier to each possible label, and that classifier can then distinguish between that particular label, and that which is "other" (the other labels). Thus, Logistic Regression was able to serve our purpose here. The data was standardized with sklearn's

preprocessing.scale function, which scales the feature measurements down around a center of zero.

Hyperparameters

A held-out test set of 10% of the full data size was chosen at the beginning, but different sizes of training/dev sets, as well as number of iterations for the Logistic Regression model, were experimented with. The following tables each show the average over 5 runs of the following values:

accuracy, as well as precision, recall, F1, and support for each humor category. The conditions under which table was created are listed above the table. This averaging was done to enable different training and dev sets each time, since they were randomly split and the dataset overall leans heavily toward being `Observational_Wordplay`.

Wordplay

Wordplay was one of the most well-represented categories in the annotated data, with 765 tweets labeled as wordplay.

Feature Engineering

To begin with, we used stemmed unigram word counts as basic features. We excluded hashtags, mentions, punctuation, and stopwords, and used NLTK's `word_tokenizer` and the Porter Stemmer to normalize the tokens. Pedersen (2017) explores the predictive power of word sense disambiguation at different context windows for whether a sentence contains a pun or not. Pedersen uses four different configurations of the `WordNet::SenseRelate::AllWords` algorithm (Pedersen and Kolhatkar (2009)) with sentence-level or trigram windows and with or without word compounding to construct four different sense assignments to the words in a sentence, then classifies the sentence based on how much these sense assignments differ between the different configurations. Pedersen's model classifies a sentence as containing a pun if there are more than two changes in sense between different configurations. Pedersen (2017)'s methods achieve some success, with

F1 scores of .83 and .80 for homographic and heterographic puns respectively, against datasets consisting of 71% puns. Pedersen's model appears to be a simple decision test based on the choice of two changes as the threshold for labeling a sentence a pun; we hypothesized that we might be able to learn the best threshold of window-based sense differences using the perceptron model. We attempted to supplement our unigram features with a similar method. Unfortunately, `WordNet::SenseRelate::AllWords` is written in Perl, but the `pywsd` package (Tan (2014)) appears to provide similar functionality. We ran the `max_similarity` algorithm across each tweet in our data, using two runs instead of four: the first with a window of the entire tweet, and the second with a window of one token before and one token after. We used this data to add two features: one with the raw count of differences between the two runs, and a second with this count normalized by tweet length in tokens. We included this second feature, which Pedersen (2017) did not discuss, in an effort to account for noise that longer tweets might accrue, with longer tweets having the potential for more differences in sense assignments.

Model

We used the Multi-Layer Perceptron classifier from `scikit-learn`, with one hidden layer and using the LBFGS solver and a maximum of 500 iterations until convergence. The LBFGS solver was chosen, per `scikit-learn`'s suggestion, to suit the small size of the dataset (Pedregosa et al. (2011)).

Experimental Settings And Results

Generalized Approach

Training size: 70% Iterations: 300 Accuracy: 0.357

Label	Precision	Recall	F1	Support
ABSD	0.3	0.007	0.014	52.8
IRNY	0	0	0	7.4
ISLT	0	0	0	21.4
OBSV	0.757	0.983	0.855	351.6
VLGR	0.612	0.095	0.163	84.2
WPLY	0.69	0.627	0.657	189.6
OTHR	0	0	0	9
NHMR	0	0	0	14.2

Training size: 80% Iterations: 300 Accuracy: 0.379

Label	Precision	Recall	F1	Support
ABSD	0	0	0	27.6
IRNY	0	0	0	4.8
ISLT	0	0	0	12.6
OBSV	0.781	0.977	0.868	242.4
VLGR	0.55	0.085	0.147	58.4
WPLY	0.677	0.686	0.677	121.4
OTHR	0	0	0	4.6
NHMR	0	0	0	11.4

Training size: 90% Iterations: 300 Accuracy: 0.37

Label	Precision	Recall	F1	Support
ABSD	0	0	0	15
IRNY	0	0	0	2.4
ISLT	0	0	0	8
OBSV	0.769	0.988	0.865	119
VLGR	0.611	0.104	0.173	26.8
WPLY	0.687	0.61	0.644	59.8
OTHR	0	0	0	4.2
NHMR	0	0	0	7.4

After seeing these results, a training size of 70% of the dataset seemed to yield the overall best results. Of note is the nonzero values for ABSD, though very low– this seems to be an indication that the smaller training set is preventing so much overfitting (likely to tweets that are OBSV and VLGR, a large part of the dataset). This is, of course, a good thing for the model in general. With the training size set at 70%, different numbers of iterations were experimented with:

Training size: 70% Iterations: 100 Accuracy: 0.361

Label	Precision	Recall	F1	Support
ABSD	0.2	0.007	0.014	48
IRNY	0	0	0	6.4
ISLT	0	0	0	23.4
OBSV	0.76	0.98	0.85	353.2
VLGR	0.701	0.134	0.225	87.8
WPLY	0.683	0.657	0.669	191
OTHR	0	0	0	6.4
NHMR	0	0	0	14.8

Training size: 70% Iterations: 200 Accuracy: 0.367

Label	Precision	Recall	F1	Support
ABSD	0.2	0.004	0.008	49.8
IRNY	0	0	0	6.8
ISLT	0	0	0	21.6
OBSV	0.763	0.985	0.86	354.6
VLGR	0.653	0.15	0.242	84.8
WPLY	0.683	0.644	0.663	192.6
OTHR	0	0	0	9.6
NHMR	0	0	0	16

Training size: 70% Iterations: 400 Accuracy: 0.366

Label	Precision	Recall	F1	Support
ABSD	0.05	0.004	0.008	48
IRNY	0	0	0	5.8
ISLT	0	0	0	23.6
OBSV	0.76	0.983	0.857	353
VLGR	0.688	0.143	0.236	88
WPLY	0.665	0.64	0.65	187.2
OTHR	0	0	0	7
NHMR	0	0	0	14

Training size: 70% Iterations: 500 Accuracy: 0.366

Label	Precision	Recall	F1	Support
ABSD	0.166	0.004	0.015	47
IRNY	0	0	0	7.8
ISLT	0	0	0	22.2
OBSV	0.758	0.984	0.856	350.4
VLGR	0.682	0.101	0.174	87.8
WPLY	0.711	0.632	0.668	196.2
OTHR	0	0	0	11.4
NHMR	0	0	0	13.4

Not much difference is seen overall between these measures, but it seems that 100 iterations produced marginally better results. Finally, using 70% training set size with 100 iterations, performance on the held-out test set was measured once:

Accuracy: 0.627

Label	Precision	Recall	F1	Support
ABSD	0	0	0	4
IRNY	0	0	0	4
ISLT	0	0	0	7
OBSV	0.97	1	0.98	167
VLGR	0	0	0	4
WPLY	0.864	0.733	0.793	139
OTHR	0	0	0	1
NHMR	0	0	0	5

Looking at all of this, it is clear that the capacity for a label to be chosen correctly is hampered by a lack of data– for those labels with low Support, and thus low occurrence in the data overall, there is not enough for the model to get to know what a tweet with that label looks like. It is for this reason that IRNY, ISLT, OTHR, and NHMR are consistently not selected, making them immeasurable and consequently set to 0. This is unsurprising with NHMR (as most tweets in the dataset participate in the joking, or are obviously attempting humor) and OTHR (there were few tweets that did not fit into Hay 1995's broad humor categories). IRNY and ISLT jokes were also simply not attempted nearly as often as OBSV and WPLY, which were both very common– this explains the artificially inflated performance on the smaller held-out test set. So much of that was

`OBSV_WPLY` that the model, already heavily biased towards those two categories, got it right much of the time by "guessing." This was hard to avoid with a small test set, which was kept small to leave sufficient data for the training and dev sets. In general, the performance is not very good.

Observational and Wordplay may be impressive individually, but this is again influenced by the fact that so much of the data lies in these categories. The addition of the `profanity-check` module for Vulgarity and Named Entity Recognition for Observational did produce noticeable improvements, as did incrementally including the two sets of general features. So, these efforts were not entirely unfruitful, but Absurdity, Irony, and Insult lie at or close to zero. These are the areas that would need the most improvement in future work.

Wordplay

The test set for this model consists of 346 tweets, with a support of 201 non-wordplay tweets and 145 wordplay tweets in the gold standard. This gives a baseline 57.8 % accuracy for a model that labels all tweets as non-wordplay.Stemmed unigram features beat baseline by about 15 percentage points:

	Precision	Recall	F1	Support	Accuracy	Kappa
Non-Wordplay	0.769	0.757	0.763	203.2	0.724	0.429
Wordplay	0.662	0.672	0.666	142.8		

Fig. 1: Multi-Layer Perceptron results, unigram features only (10 trials)

Including the word sense features yielded no significant increases in model performance over 10 trials on the same test set, in fact showing slight drops in performance:

	Precision	Recall	F1	Support	Accuracy	Kappa
Non-Wordplay	0.710	0.776	0.739	183.8	0.712	0.416
Wordplay	0.715	0.640	0.673	162.2		

Fig. 2: Multi-Layer Perceptron results incl. word sense features (10 trials)

The slight decrease in performance here likely indicates that these features are only contributing noise to the data. Note also that the support is skewed towards wordplay versus the gold standard, suggesting that the word sense features are overpredicting the wordplay label.

Conclusions and Future Work

Generalized Approach

If we were to continue this project using this method of mostly general features, proper word intensity scores should be used, as they are a rating on a scale as opposed to SentiWordNet's proportions of objectivity, negativity, and positivity. Additionally, the named entity recognition could be improved by finding the closest matching names of people/works of art/etc. (perhaps via a Google search and identification of Wikipedia articles on the named entity) and calculating an edit distance between the original name and the name as it is presented in the tweet. Such a feature would likely help identify puns and other wordplay on those named entities, rather than the current system of simply recognizing whether or not something is a named entity. In the same

vein, it was hoped that analysis of word frequency would detect Absurdity, but that did not turn out to be the case. A better feature set to identify the difficult category of Absurdity, as well as features for Irony and Insult humor, would be required, since the general set of features are not able to distinguish those labels as we had hoped. Representing the words as embeddings and calculating distance between them might be helpful in this regard. Insult specifically may benefit from improved intensity scores, since strong words are often lobbed at the victim of an insult, while Irony would require further research regarding its inner composition.

Wordplay

With stemmed unigram features alone, our classifier achieved around 72% accuracy on average, with a Cohen's kappa against the gold standard of 0.429. This suggests that in this dataset, there is some significance to the vocabulary of a tweet as an indicator for whether it involves wordplay.

The inclusion of the WordNet sense features that we created did not improve our results. There are a number of differences between the task Pedersen (2017) was targeting and our task: for example, the corpus Pedersen's model was applied to consisted of full sentences, whereas most of the tweets in the HashtagWars corpus are only a few tokens long and often consist of only a single noun phrase. `pywsd`'s `max_similarity` algorithm seems to perform much better on full sentences than short phrases. Additionally, we do not know how our method for computing sense assignments with 'pywsd' compares to Pedersen and Kolhatkar (2009)'s algorithm. Based on our experimentation, it seems surprising that they had confidence in the ability of just two tokens of context to reliably construct a sense assignment that is similar enough to the full-sentence-window run in non-pun contexts to enable their method to work at all, especially across four different configurations.

References

Hay, Jennifer (1995), *Gender and Humour: Beyond a Joke*. Master's thesis, Victoria University of Wellington.

Mahajan, Rutal and Mukesh Zaveri (2017), "SVNIT @ SemEval 2017 task-6: Learning a sense of humor using supervised approach." In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 411–415, Association for Computational Linguistics, Vancouver, Canada, URL `https://www.aclweb.org/anthology/S17-2069`.

Pedersen, Ted (2017), "Duluth at SemEval-2017 task 7 : Puns upon a midnight dreary, lexical semantics for the weak and weary." In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 416–420, Association for Computational Linguistics, Vancouver, Canada, URL `https://www.aclweb.org/anthology/S17-2070`.

Pedersen, Ted and Varada Kolhatkar (2009), "WordNet::SenseRelate::AllWords - a broad coverage word sense tagger that maximizes semantic relatedness." In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Demonstration Session*, 17–20, Association for Computational Linguistics, Boulder, Colorado, URL `https://www.aclweb.org/anthology/N09-5005`.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), "Scikit-learn: Machine learning in Python." *Journal of Machine Learning Research*, 12, 2825–2830.

Raz, Yishay (2012), "Automatic humor classification on twitter." In *Proceedings of the NAACL HLT 2012 Student Research Workshop*, 66–70, Association for Computational Linguistics, Montréal, Canada, URL `https://www.aclweb.org/anthology/N12-2012`.

Tan, Liling (2014), "Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software]." https://github.com/alvations/pywsd.

Zhou, Victor (????), "profanity-check." URL `pypi.org/project/profanity-check/`.