Can we (artificially) understand Seinfeld's humor?

Dan Kufra (30219082-2 dank) [dan.kufra@mail.huji.ac.il]
Yuval Jacoby (30224707-7 yuvalja) [yuval.jacoby@mail.huji.ac.il]
Alon Netser (31160253-6, alonnetser) [alon.netser@mail.huji.ac.il]

March 1, 2019

Contents

Abstract

Who among us isn't familiar with the hilarious sitcom "Seinfeld"? We've all laughed from Jerry's stand-up scenes (YouTube), George Costanza's ridiculous behavior (YouTube), Kramer's facial expressions and bursting into the room (YouTube), and Elaine pushing everyone away and shouting "Get Out!" (YouTube).

The main question we ask is - can we build a model that understands this humor as we do?

1 Problem Description

We want to build a model that "understands humor", i.e. given the text of a scene from Seinfeld, will predict the funniness of each sentence.

First, we validate the data. We find certain flaws and inaccuracies, and think about how to deal with them. Second, we analyze the data and visualize its content. Then, we visualize the connections between the different characters in the show, and build an interactive graph describing these relations.

1.1 Difficulties

The type of problem we are handling is very difficult, we will try to break them into groups:

1.1.1 Computational Humor detector

We use humor all the time, but it's still hard to explain why something is funny. Computational understanding of humor is even harder. There are plenty of works in this field; for example [2, 3, 5].

One of the problems with detecting humor is the subjectiveness of the term "funny" i.e. the absence of ground truth.

In this task, we will try to predict the Seinfeld writer's humor, in this case we do have labels, thought this work we use funny as funny be Seinfeld writers.

1.1.2 Understanding Text

In order to understand humor in text we need to understand the text where we have ambiguity of words, dynamic language etc..

Moreover, context and timing of text are really important for understanding humor. When we watch an episode of Seinfeld we understand the overall topic, and remember events that occurred previously in the episode (and even in other episodes). In order to make the computer understand the humor as well, we must use a model that is capable of understanding context and "remembering" important events from the past.

1.1.3 Funny is not taken only from the text

When we watch Seinfeld and laugh, it is not only because of the text. Most of the times it is affected from the tone, and more generally the whole scene's video. For example when Kramer enters the room demonstratively "in a funny way". These aspects are not always obvious in the text itself.

Some works try to predict funninesses in sitcoms, such as "The Big Bang Theory" [4]/

2 Data

2.1 Introduction

The data we use is basically Seinfeld's subtitles, coupled with the speaking character and timing - of the sentences said and the laugh-tracks (if funny).

We downloaded the dataset from GitHub, and we thank Ran Yad Shalom and Yoav Golberg for the great dataset they built [1].

2.2 Specs

The dataset contains 96 humor annotated "Seinfeld" screenplays, along with the timing of the laughter and the timing of the dialog.

R. Y. Shalom and Y. Golberg got the subtitles from opensubtitles.org, the scripts from seinology.com and used the audio tracks to extract the exact timing of the laugh-tracks. Furthermore, they used quite sophisticated techniques to align the subtitles with the exact timing, and attach the speaking character for every sentence using the scripts. You can read about it further in [1].

Due to the technique they used to build the data (using the fact that the dialogs were recorded in mono, and the laugh-track in stereo) we have the episodes starting at season 4 episode 6. This is because the previous episodes were not recorded this way.

There are 46,497 sentences in total, associated with several properties:

- 'character': The speaking character.
- 'txt': The text.
- 'start': Start time (in seconds).
- 'end': End time (in seconds).
- 'is_funny': Whether a laugh occurred after this sentence or not.
- 'laugh_time': If this sentence is funny, this is the timing of the laugh (in seconds). Note that if the sentence was not funny, this is set to NaN.
- Various meta-data about the episode, such as 'season' (season's number), 'episode_num', 'episode_name', 'total_lines', 'global_episode_num' (in the whole dataset).
- Useful features of the sentence, such as 'num_words', 'length' (in seconds), 'line_num' (in the episode), 'avg_word_length' (in letters).

For example, here are 3 lines from our dataset (meta-data such as season, episode, etc omitted).

character	**text**	**start**	**end**	**is_funny**	**laugh_time**
SUSAN	I told you to take the offer.	199.003	201.469	F	
GEORGE	Look, I had nothing to do with this. It wasn't my decision.	201.539	205.7	F	
GEORGE	It was Jerry. Jerry told me. I'm the creative guy.	206.044	209.341	T	208.3

Out of the 46,497 there are

2.3 Data Validation - no dataset is perfect in our world...

Remark. This section was done in an interactive Jupyter Notebook named 'Data_Cleaning.ipynb'. You are more than welcome to take a look!

The first task was analyzing the dataset and validate it. We watched several episodes and looked at the dataset at the same time.

We saw that the text is pretty accurate, except minor glitches such as, "You met her in the supermarket. How did you do that?" was shorten into "You met her in the supermarket. How?".

The timing of the talking are also pretty accurate, and by reading the paper of Ran Yad-Shalom [1] who created this dataset he addressed this issue specifically and payed extra attention to take several subtitles and choose the one that is best aligned with the audio.

2.3.1 The first flaw: the laugh-tracks

The laugh track is in middle of a sentences and sometimes it is during multiple sentences.

We treat a sentence as funny if there was a laughter during the sentence (sentence start <= laugh time <= sentence end).

2.3.2 The second flaw: the speaking character is sometime mislabeled.

While visualizing the data we saw a weird phenomenon (fig 1), the histogram of the amount of sentences in a row has a very long tail, with up to 31 sentences in a row.

While there are scenarios where this is possible (a phone call where we hear only one side, or a long monologue), we decided to further investigate and watched multiple scenes where this happens, while doing so we did see that in many situations this is a mislabel.

To do this, we collected all these cases, and analyzed them statistically. We saw that the mean sentences-in-a-row is around 1, and that 99% of the times it is below 8.

Our solution was to mark these speaking characters as 'unknown', but keep the other attributes (such as text, duration, funniness, etc).

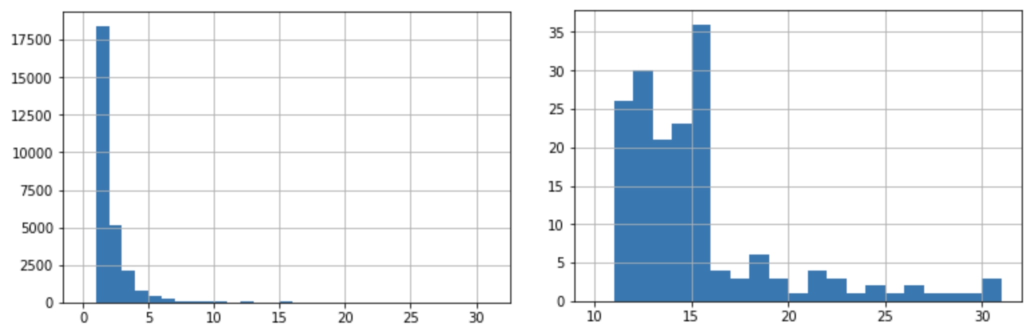


figure 1 - (Left) Number of sentences in-a-row.
(Right) Number of sentences in-a-row, bigger then 10 (the "tail").

2.4 Data Visualizations

Remark. This section was done in an interactive Jupyter Notebook named 'Visualizations.ipynb'. You are more than welcome to take a look!

We tried to visualize several aspects of the data, in order to get more insight about it and get ideas about how to solve it.

2.4.1 Finding Meaningful Features for a Sentence

We wanted to see if there are differences between funny and not-funny sentences in several aspects, such as length (in seconds), number of words in the sentence, speech-rate (words per second), etc. We saw that the distributions of the length / #words is slightly different between funny and not-funny sentences. However, these differences are not significant, as the standard deviations are also pretty high. We found that the length in seconds and the length in #word behave different, and the speech-rate seems also like a good feature (see the figure below).

Our conclusions were to add these features to the sentence. We saw that the results improved when we gave them these features.

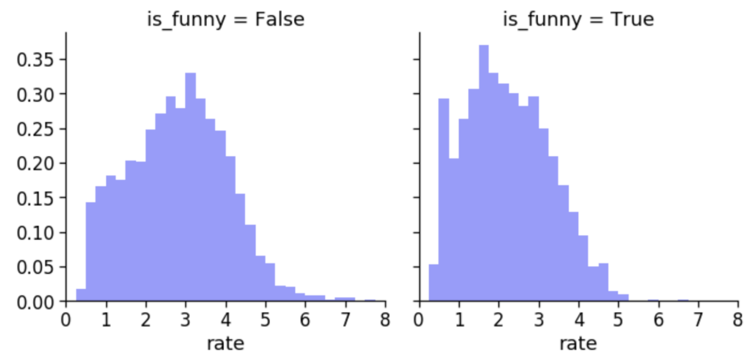


figure 2 - Elaine's speech-rate

2.4.2 Word-Clouds

Intuitively, each character tends to use different vocabulary, we tried to see if that also holds for funny/not funny sentences.

To do so we removed very frequent English words (using gensim corpus).

First, we found that the main characters have a variety of words which they all use, for example: *yeah, oh.*

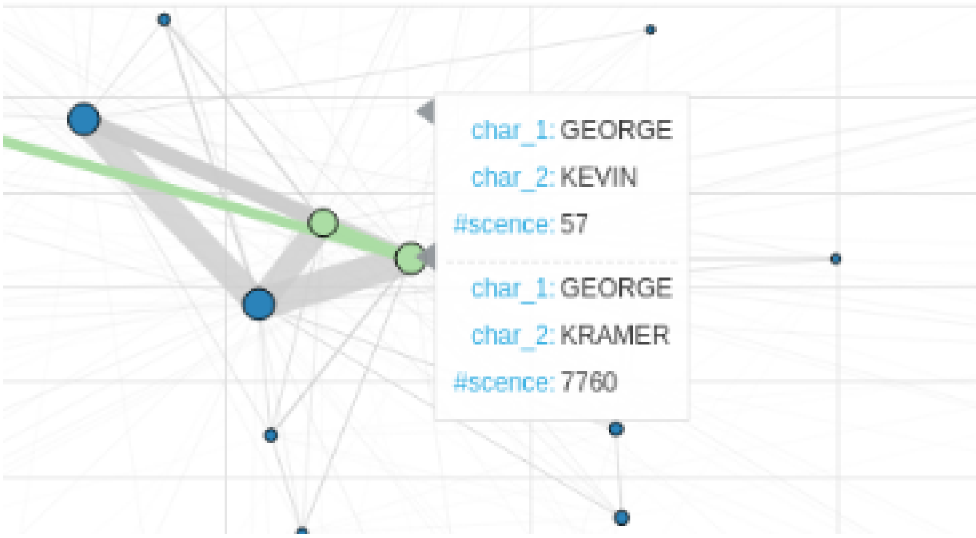
And also a couple of distinct words such as: Jerry - *car*, Kramer - *Newman.*

Further we wanted to see if characters use different words in funny and non funny sentences.

In order to get significant words, we filtered out words that are common in both types i.e. we used only words that: $\frac{\#funny}{\#total} > 0.5$, in figure-3 we can see Jerry's word clouds, for example when Jerry talks about dating it's usually funny.

2.4.3 Characters Graph

Another nice visualization was the characters-graph, containing the different characters and their connections. Each edge's weight is proportional to the amount of times the characters spoke to each other. One can see perfectly the "community" of the 4 main characters (Jerry, George, Kramer and Elaine), as well as other secondary characters (such as Jerry's parents Morty and Helen), this graph is interactive and for full experience please open the notebook.



3 Experiments

In our attempt to succeed at this prediction task we implemented a number models. These can be split into bag-of-word type models and sequence type models.

Bag-Of-Words type:

In this type of model we represent each sentence as a one-hot encoding of the sentence in a vector the length of all the words in our corpus. For example, for the sentence "He's a bubble boy!" we would have an 1 in the corresponding index of each word. For the sentence "Yada yada yada" we would have the index corresponding to the word "Yada" be equal to 1. These types of models do not preserve order of the words and as such we did not expect them to be able to learn much. Nonetheless, we implemented both logistic regression (with binary bag of words) and a multi-layer perceptron model (with tf-idf bag of words and ngrams).

Sequence type:

In this type of model we represent each sentence as a series of words, where each word is a one-hot encoding of that word in a vector the length of all the words (as in the bag-of words model). The difference is that we **do** preserve the order of the words. We felt these type of models would more accurately represent our task since we could not find any major correlation between funniness and certain words.

In addition, after our initial data analysis we saw that many of the high-level features of the sentence can be indicative of its "funniness". So in each of these models we also created version where the following features are inputted at some stage:

- An encoding of which character is speaking (For the characters: Jerry, George, Elaine, Kramer, Morty, Helen, Frank, Estelle, Newman)
- The start time of the sentence.
- The length of the sentence in time.
- The amount of words in the sentence.
- The rate of speaking (length/number of words).
- The average word length of the sentence.

The first two models implemented worked per sentence. Meaning they did not use information from sentences that are from the same scene/episode and may be relevant. These models are:

Long Short Term Memory model:

Given a sentence each word is put through a Glove embedding layer to a dimension of 100. These embeddings are then put through a Bidirectional LSTM with an output of 128 units. We use an attention mechanism on the series of outputs. We then either concatenated the high level features or didn't (depending on the experiment). Then after a few more Fully Connected layers we output a measure of funniness.

We also added another output directly after the attention mechanism (before the concatenation with high level features) in order to ensure the loss propogates in some way back to the LSTM cells. Our final loss is a weighted binary cross-entropy loss of these two outputs.

Convolutional Neural Network model:

Given a sentence each word is put through a Glove embedding layer to a dimension of 100. This is then passed through 3 blocks of the following:

- Seperable 1D Convolution
- Seperable 1D Convolution
- 1D Max Pooling

Then a 1 dimensional Average Pooling layer and a Fully Connected layer. Then depending on the experiment, a concatenation of the high level features and another Fully Connected layer for the output. We then apply a binary cross-entropy loss on the two outputs.

The final model we implemented did make use of the additonal information of other relevant sentences:

Long Short Term Memory Multi-Sentence model:

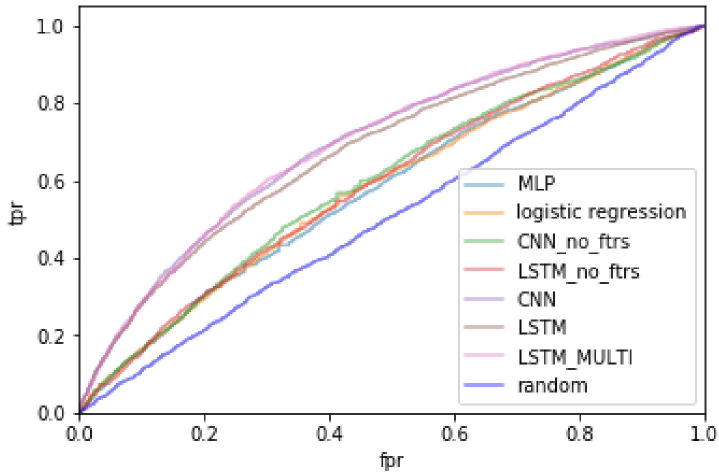
In this model each example is actually a series of sentences in an episode. Each sentence on its own is represented in the same way as in the previous two models. We then pass each sentence through a CNN model as described before. But then instead of predicting one output for each sentence, we pass all the sentences (after the convolution model) through a Bidirectional LSTM. We then concatenated each sentence's high level features and go through a final Fully Connected layer before predicting the "funniness" per sentence and applying a weighted binary cross-entropy loss on the two outputs.

All the model graphs are included in the appendix.

4 Examining the Results

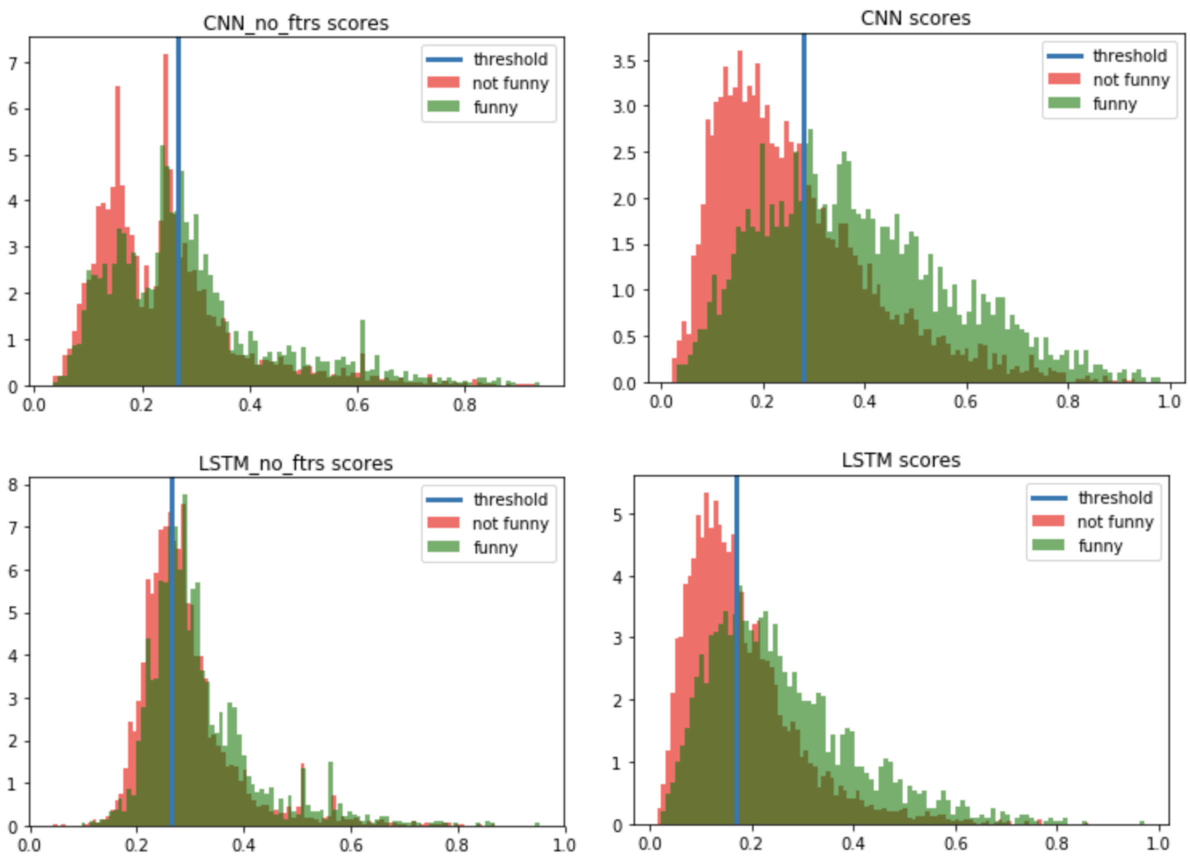
Remark. This section was done in an interactive Jupyter Notebook named 'Analyze_Resulsts.ipynb'. You are more than welcome to take a look!

After we finished training the different models we built, we turned to analyze their performance. We looked visually at some of the results, and plotted several measures.



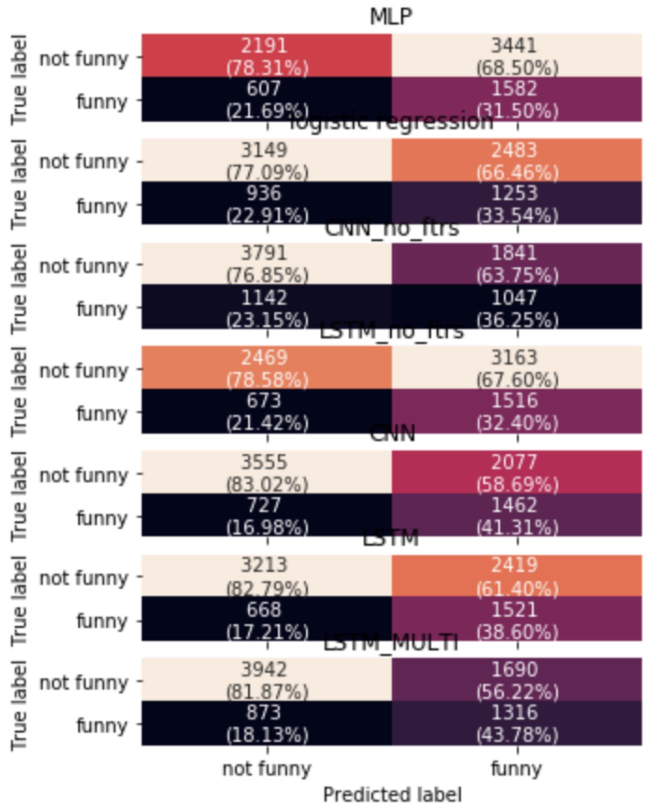
ROC-curves

We saw that the features we added helped the models greatly. Here are the scores for the CNN and LSTM models, with/without the additional features.



Scores with/without the additional features.

Numerically, here are the confusion matrices for the different models:



Confusion matrices

It's not perfect, but it definitely learned something. We must remember that our data is quite noisy, so nothing can do it perfectly.

We tried to split the 4 main characters and see if the performance are different when we look at certain character. We found that there are minor changes, but overall they're all quite the same. You can see for yourself in the notebook.

Furthermore, we wanted to look at some of the results visually.

Here is the most severe false-positive (i.e. the model thought it's funny but it doesn't). We also add some context (5 sentences before the allegedly 'funny' sentence).

```
LSTM score is 0.86
character                                                       txt  is_funny
   ELAINE                                    "I'm very impressed"?     False
   ELAINE            You mean "pressed"  because it's a dry cleaner?     False
    JERRY                                Yeah, see? That's why I hate it.     False
    JERRY                                    So come on, you wanna go?     False
   ELAINE  Well, what about the sleeping  arrangements in the cabin?     False
    JERRY            Well... Same bed,  and underwear and a T-shirt.     False
```

"Severe" FP

Here are the severe false-negatives (i.e. the model thought it is not funny but it is):

```
LSTM score is 0.03
character                                    txt  is_funny
     ALAN      I hope we can get  past all this.     False
     ALAN              Oh, past? We're way past.     False
     ALAN                So you have a big head.     False
   ELAINE                               So what?      True
     ALAN  Goes well with the bump  in your nose.     False
   ELAINE                                  What?      True

LSTM score is 0.031
character                                                      txt  is_funny
   ELAINE                                      I spoke to Alan.     False
   ELAINE     You know, I told him  I didn't wanna see him anymore.     False
   ELAINE                                Called me "big head."      True
    JERRY                                  [SCOFFS]  Big head?     False
    JERRY                            It's almost a compliment.      True
   ELAINE  It's one of the nicest things  anyone's ever said to me.      True
```

"Severe" FN

These mistakes don't seem that severe. By looking at it (without checking the label) one can think it's label is the opposite. It demonstrates how difficult this task is.

5 Future Work

TODO

6 Conclusion

TODO

References

[1] R. Yad-Shalom and Y. Goldberg, The Seinfeld Corpus: A Method for Generating A Humor Annotated Corpus and An Analysis of That Corpus. Computer Science Department, Bar-Ilan University, Israel, 2017.

[2] R. Mihalcea and S. Pulman. Characterizing humour: An exploration of features in humorous texts. In Computational Linguistics and Intelligent Text Processing, pages 337–347. Springer, 2007

[3] D. Shahaf, E. Horvitz, and R. Mankoff. Inside jokes: Identifying humorous cartoon captions. In SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015.

[4] Bertero, D., Fung, P.: A long short-term memory framework for predicting humor in dialogues. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (2016)

[5] Aditya Joshi, Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, and Mark Carman. 2016. Are word embedding-based features useful for sarcasm detection? In Conference on Empirical Methods in Natural Language Processing (EMNLP)