Funny Or Not

Humorous content detection in user reviews

By Aibek Uraimov

Problem Identification

Humorous content detection in user reviews

Funny Or Not Problem Statement

Detect humorous content in user reviews of businesses, using Yelp dataset on restaurant reviews, with at least 15,000 observations, for training a prediction model with accuracy score above 65%, and f1-score of at least 0.70 for both classes.

Context

I always wondered what makes a joke funny. Obviously, a context.

It would be nice to have a machine be aware of and 'understand' context and then crack jokes all day. However, that is a goal for another project – in this one, I am interested in testing a basic machine learning tool to find humorous text bits online.

Yelp platform has aggregated a massive database of customer reviews on businesses for a wide range of categories. My particular interest lies in detecting funny element in customer reviews.

Criteria for success

The NLP model will be able to take text from unlabeled Yelp reviews or arbitrary text as input and detect whether they are 'funny' or not.

Criteria for the success rate is any results > 65%, i.e., better than just a guess. It is set low for this project assignment, as detecting humor in a human language is a tremendous task and realistically, would require a lot more resources than are available for this academic exercise.

The criterion for success is for at least one of the models to reach a prediction accuracy score above 65% and F1 score of at least 0.70 for both classes ('Funny' and 'Not Funny').

Scope of solution space

The goal is to train classification models on a sampled Yelp dataset and feed it with arbitrary review texts for it to detect funniness.

Scope of solution space

The scope of this project would be a classification model the content of reviews – limited to restaurants.

Why just restaurants? Here is an initial theory – restaurants are most widely represented in Yelp and probably have the most polarized reviews and consequently, be rich in humorous remarks.

Another hypothesis I would like to check is whether there is a correlation between the rating of a business and frequency of funny content in the reviews.

Constraints within solution space

- Humor is subjective and what might be labelled 'funny' by one person, may not be considered so at all by another – hence, we might be getting some True Positives in a review instance, which are actually False Positives if you ask an alternative opinion. This might be a challenge in terms of integrity of the labelling.
- Much of hilarious written content that goes viral does so because there is already a known context by the masses. So, even if a text appears completely neutral, it becomes very funny, once combined with an appropriate context. Our model would not 'know' no such context, so it will struggle a lot figuring out why certain review texts are getting a lot of 'funny' labels with no apparent statistical indicators
- Finally, quantitative constraints included the availability computing resources aka a laptop

Funny Or Not Problem Statement

Detect humorous content in user reviews of businesses, using Yelp dataset on restaurant reviews, with at least 15,000 observations, for training a prediction model with accuracy score above 65%, and f1-score of at least 0.70 for both classes.

Stakeholders to provide key insight

Key stakeholder and Subject Matter collaborators (aka Audience, or Social Media Content Consumers):

- Aibek Uraimov – aspiring Data Scientist student at a SpringBoard Data Science Track
- General Public – subject matter experts in detecting sarcasm and appreciating some straight up good writing styles

Key data sources

Yelp Open Dataset Link: https://www.yelp.com/dataset

Yelp datasets:

- Business – 121MB JSON file
- Check-ins – 389MB JSON file
- Reviews – 6,774MB JSON file
- Tip – 225MB JSON file
- User – 3,598MB JSON file

Within these datasets, the columns of interest were:

- 'categories' - to filter on 'Restaurants' only
- 'starts' – business rating
- 'rating' – review rating
- 'review_count' – measure of a public interest
- 'funny' – the most important column, massive "labeling" for supervised learning

The Data

Humorous content detection in user reviews

Data Sourcing

Raw Data

The reviews dataset was close to 7Gb, so I had to search ways to load it to Jupyter Notebook without breaking my computer, which is what I did by following this super helpful post.

Basically, what you do is load JSON data in manageable chunks of 1MM lines using pandas' *pd.read_json* and indicating the *chuncksize*. I then merged the Business and Reviews datasets and saved it as '.cvs' file.

Initial Data Preparation

I have extracted the 'Restaurants' data and had to reduce the dataset even further:

- my initial attempts of feeding it into a model resulted in 2+ hours of my computer spinning the fans and significantly ruining my excitement for learning
- I reduced the still gigantic file to the last 7 years of reviews and kept only 1MM lines

Reducing Data Size

I dropped columns that did not add anything for the purpose, such as business_id, address, latitude, longitude, is_open, attributes, categories, hours, user_id, date

Data Wrangling

Cleaning

- The datasets had negligible number of missing values – I simply dropped them.

Categorical Labels

The restaurants have been given star ratings, ranging from 1 to 5 with a 0.5 increments.

I reassigned them to either ‘Low Rating’ or ‘High Rating’ instead: 1 through 3.5 got to be ‘Low Rating’ and 4 through 5 were now ‘High Rating’ establishments.

The ‘is funny’ target variable was represented by a column of 0 or 1 values – I assigned these ‘Not Funny’ labels for 0s and ‘Funny’ for 1s.

Exploratory Data Analysis

Humorous content detection in user reviews

EDA

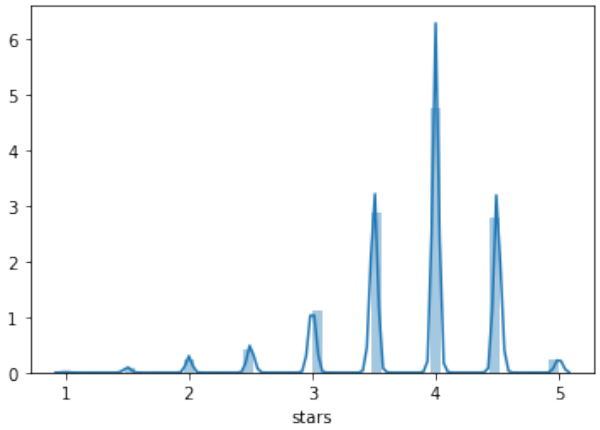
Target Variable

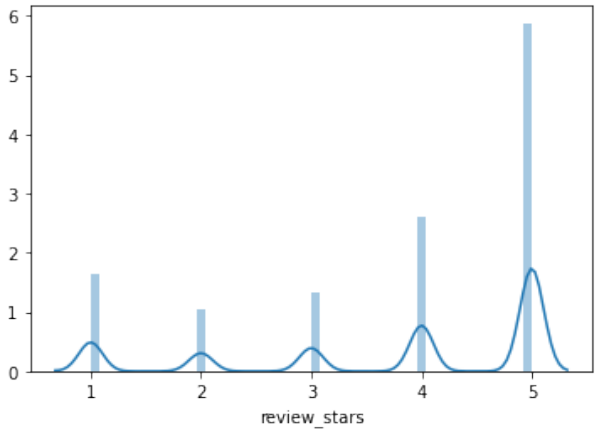
This is a classification problem – I added a 'Funny Meter' field into the final source dataframe

- **Funny Meter is the target variable** to be predicted in the modeling

Distributions

Here is a distribution of Business Ratings and Review Ratings next to each other:





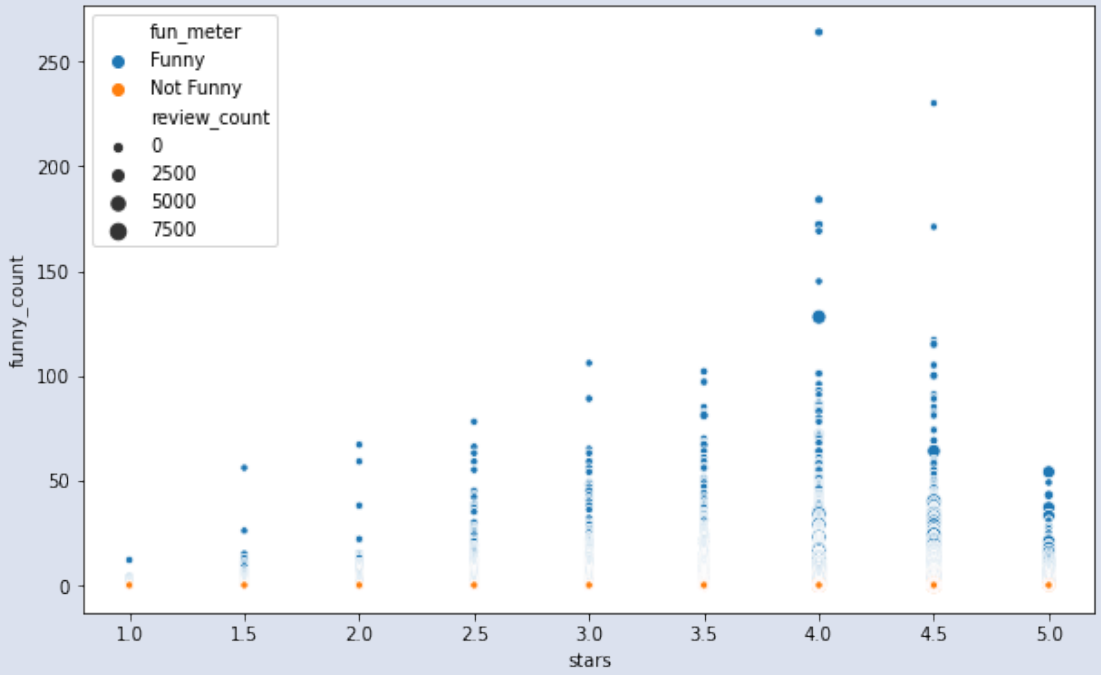
Most business ratings have 3.5-4.5 stars, indicating that thriving businesses stay so, because they provide above average product.

Review ratings pertain more to the writing ability of the reviewer than a business.

EDA

Correlation – Business Rating to Funny Review Frequency

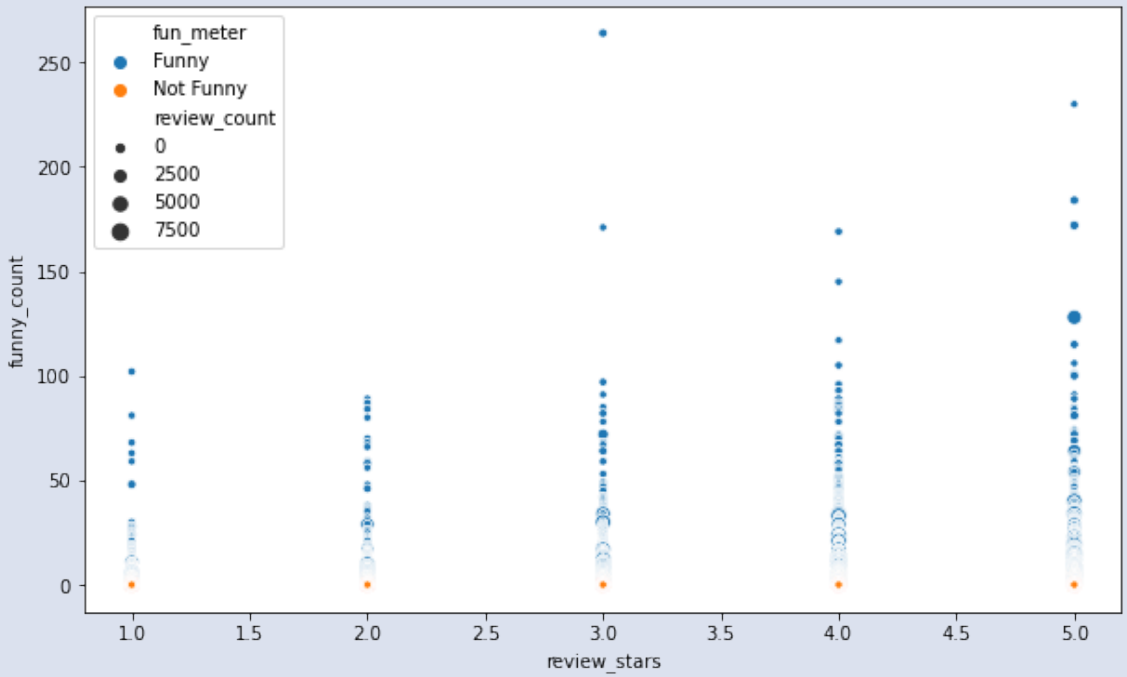
- although the count of funny reviews is consistent with the count of overall reviews, it looks like the users like to crack jokes about their bad experiences equally with the better ones



EDA

Correlation – Review Rating to Funny Review Frequency

- Wow, will you look at that – the folks are happy to laugh at anybody's expense it looks like, be it someone rated as 'a good' reviewer or 'a bad' reviewer alike



Pre-processing and training

Humorous content detection in user reviews

Pre-Processing and Training

Is data balanced?

Before pre-processing work, let's check if the dataset is balanced:

```
funny_count = len(df.loc[df.fun_meter=='Funny'])

not_funny_count = len(df.loc[df.fun_meter=='Not Funny'])
```

Output:

```
The ratio of Funny to Not Funny in the dataset is Funny:
79672  to Not Funny:  493966 or  16.13 % to 100%
```

Nope, not balanced – this would lead to skewed results, so rebalancing:

- split the dataset into 'Funny' and 'Not Funny' dataframes
- balanced the size of the disproportionally larger 'Not Funny' dataframe to equal that of the 'Funny' one
- concatenated them back

Final dataframe

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 15934 entries, 128037 to 3871
Data columns (total 8 columns):

 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   stars          15934 non-null  float64
 1   review_count   15934 non-null  int64
 2   review_stars   15934 non-null  int64
 3   is_funny       15934 non-null  int64
 4   funny_count    15934 non-null  int64
 5   text           15934 non-null  object
 6   rating         15934 non-null  object
 7   fun_meter      15934 non-null  object

dtypes: float64(1), int64(4), object(3)
memory usage: 1.1+ MB
```

Pre-Processing and Training

NLP Analysis with spaCy

Using spaCy open-source library with *en_core_web_sm model*, and *scattertext* tool, I set up a collection of texts from the *df_reviews* dataframe and obtained **term frequency** and **scaled f-score values**, which were then used to create 2 dataframes:

- *funny_reviews top 5*
- *meh_reviews top 5*

Each dataframe contained 405,272 entries.

The following terms were listed with highest Funny/Not Funny scores:

Term Frequency

funny_reviews dataframe – top 5 by Funny_Score

	term	Not Funny freq	Funny freq	Funny_Score	Not_Funny_Score
483	chinese	29	360	1.00	0.00
2483	chinese food	6	85	0.99	0.01
1959	airport	17	106	0.98	0.02
2006	looks like	14	103	0.98	0.02
1126	pot	32	178	0.98	0.02

meh_reviews dataframe – top 5 by Not_Funny_Score

	term	Not Funny freq	Funny freq	Funny_Score	Not_Funny_Score
667	books	181	39	0.00	1.00
1419	great atmosphere	91	22	0.00	1.00
935	lobster roll	135	42	0.01	0.99
1578	screen door	83	28	0.01	0.99
1236	oyster	103	37	0.01	0.99

Pre-Processing and Training

Models

Four models have been created for this project:

1. Model 1 – Linear SVM Classifier
2. Model 2 – Random Forest Classifier (Version A)
3. Model 3 – Random Forest Classifier (Version B)
4. Model 4 – Linear Classifier with SGD learning

Model Scenarios

If only there was a way to pit them against each other on an equal basis… Wait, there is!
I prepared four arbitrary text input examples to be used for testing each of the four models:

1. Example 1: `my_review_1 = ["The place is truly a touristic trap dump! It looks like a restaurant, but it is not!"]`
2. Example 2: `my_review_2 = ["I told the owner that mochi and doughnuts were speaking to me the words of wisdom"]`
3. Example 3: `my_review_3 = ["Can't even explain how much I love this place"]`
4. Example 4: `my_review_4 = ["This restaurant is not very good. Giving this 3.5 stars and not coming back."]`

Modeling Results and Analysis

Humorous content detection in user reviews

Modeling Results and Analysis

Model 1 – Linear SVM Classifier

The first model was Support Vector Machine, a liner model for classification problems:

Model 1 performance evaluation:

Accuracy score:

```
clf_svm.score(x_test_vector, y_test)
```

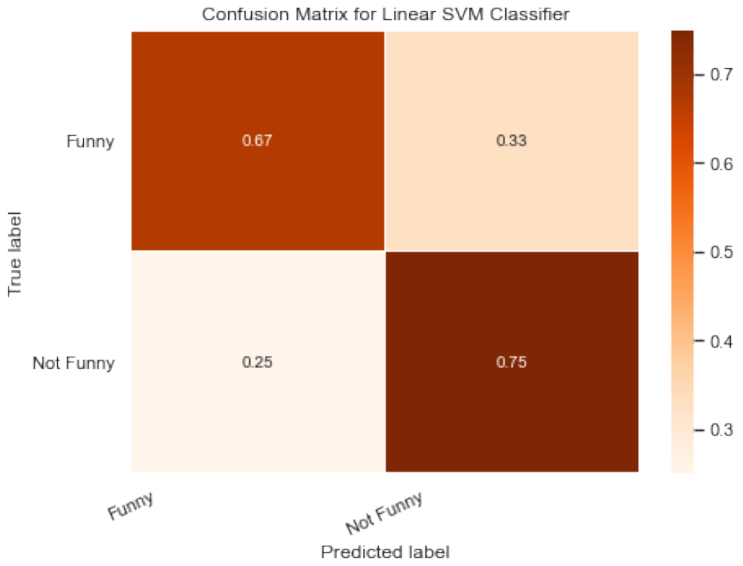
```
0.7105923694779116
```

And f1-score:

```
f1_score(y_test, clf_svm.predict(x_test_vector), average=None)
```

```
array([0.69745474, 0.72263652])
```

Model 1 – Confusion Matrix



Modeling Results and Analysis

Model 2 – Random Forest Classifier (Version A)

The second model I used was Random Forest Classifier, with the following parameters passed in: *bootstrap=False*, *criterion='entropy'*, and *n_estimators=100*

Model 2 performance evaluation:

Accuracy score:

```
classifier_rf.score(x_test, y_test)
```

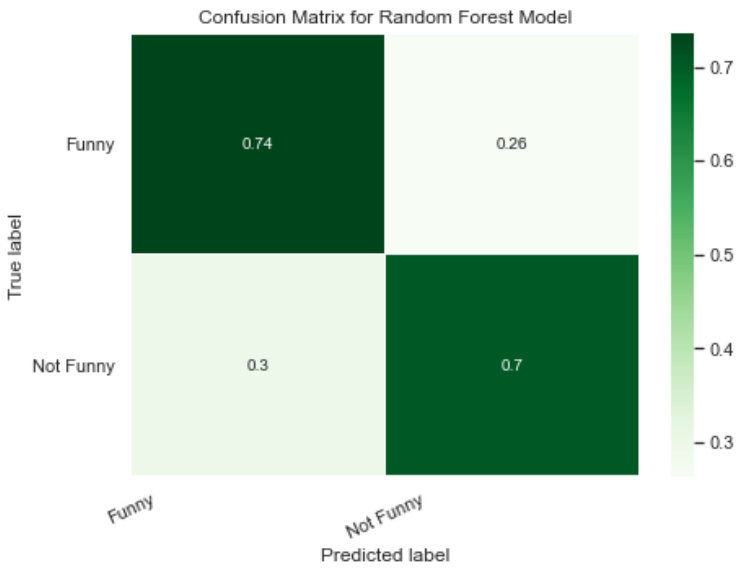
```
0.7201305220883534
```

f1-score:

```
f1_score(y_test, classifier_rf.predict(x_test), average=None)
```

```
array([0.72325639, 0.71693323])
```

Model 2 – Confusion Matrix



Modeling Results and Analysis

Model 3 – Random Forest Classifier (Version B)

The third model was another Random Forest Classifier – this time with default parameters but passing the following parameters into the ***tfidf*** vectorizer in the pipeline: *stop_words='english', ngram_range=(1,1)*.

Model 3 performance evaluation:

Accuracy score:

```
pipe_rfc.score(x_test, y_test)
```

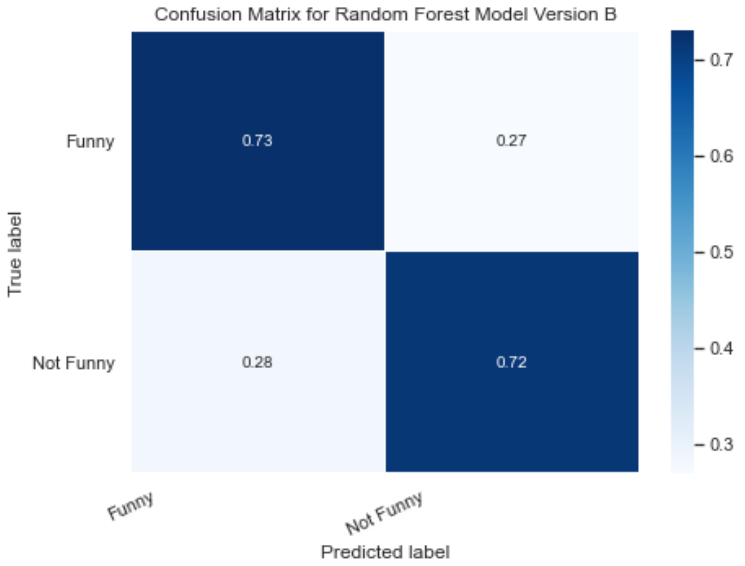
```
0.7238955823293173
```

f1-score:

```
f1_score(y_test, pipe_rfc.predict(x_test), average=None)
```

```
array([0.72458688, 0.72320081])
```

Model 3 – Confusion Matrix



Modeling Results and Analysis

Model 4 – Linear SGD Classifier

The fourth and final model was a linear classifier with ***stochastic gradient descent (SGD)*** learning.

Yep, I know – the thing does sound pretty cool.

Model 4 performance evaluation:

Accuracy score:

```
pipe_sgd.score(x_test, y_test)
```

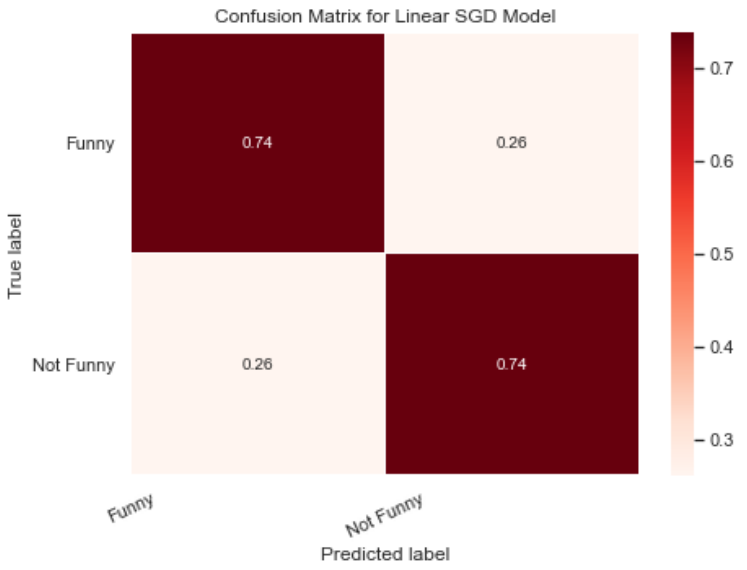
```
0.7377008032128514
```

f1-score:

```
f1_score(y_test, pipe_sgd.predict(x_test), average=None)
```

```
array([0.73644388, 0.73894579])
```

Model 4 – Confusion Matrix



Modeling Results and Analysis

Summary of model performances

Model 1:		
	Accuracy Score:	0.7105923694779116
	f1_score:	[0.69745474, 0.72263652]
Model 2:		
	Accuracy Score:	0.7201305220883534
	f1_score:	[0.72325639, 0.71693323]
Model 3:		
	Accuracy Score:	0.7238955823293173
	f1_score:	[0.72458688, 0.72320081]
Model 4:		
	Accuracy Score:	0.7377008032128514
	f1_score:	[0.73644388, 0.73894579]

Model Selection

Based on the Confusion Matrix metrics, Model 4 is selected as the best model for running prediction scenarios.

Summary and Conclusion

Humorous content detection in user reviews

Summary and Conclusion

Summary

- Here's the summary of outcome of feeding 4 Test Scenarios with review text inputs into the models:

	Input	Outcome
Model 1 - Linear SVM Classifier	Example 1	Funny
	Example 2	Not Funny
	Example 3	Not Funny
	Example 4	Not Funny
Model 2 - Random Forest Classifier (Version A)	Example 1	Not Funny
	Example 2	Not Funny
	Example 3	Not Funny
	Example 4	Not Funny
Model 3 - Random Forest Classifier (Version B)	Example 1	Funny
	Example 2	Not Funny
	Example 3	Not Funny
	Example 4	Not Funny
Model 4 - Linear SGD Classifier	Example 1	Funny
	Example 2	Funny
	Example 3	Not Funny
	Example 4	Not Funny

Summary and Conclusion

Conclusion

Judging by performance evaluations, we would be better off sticking with Model 4 - after all it seems to have the best sense of humor!!!

Also, let us not forget that the 'funniness' of the texts is always going to be subjective - at least by us, the humans.

Recommendations

I am definitely going continue work in exploring ways to detect funniness around us – I think it is key to getting close to unleashing the power of true AI. I suspect it will take a slightly different approach than just applying statistics to text.

As for the potential applications stemming from this exercise, here are a few: 1 – bots for online games (quests) that require ‘conversations’ with the gamer, 2 – platform for detecting and aggregating written content with high chances of going viral on social media, 3 – research on building the true AI.

Further Work

This is the end of the presentation and here is a brain teaser as a parting offering:

- people react strongly to depictions of potentially hazardous/critical/dangerous situations – especially if funny element is present
- Why? Because funny implies a more or less harmless, or at least, bearable to watch resolution of those situations
- Combination of “situation” + “resolution” => ”funny” is basically a highly valuable lesson that humans may use, should they find themselves in a similar emulation, I mean, situation
- Valuable lessons == high interest => viral content sharing

Thank you for your attention!