Robbins, Liz and Nadia Rodriguez. (2017). The Gang Murders in the Suburbs. The New York Times. https://www.nytimes.com/2017/07/12/nyregion/ms-13-murders-long-island.html?_r=0.

Rodriguez, Ana Patricia. (2009). Dividing the Isthmus: Central American Transnational Histories, Literatures, and Cultures. 168-171. 176. 5.

Salgado, Yesika. (2017). Corázon, 4. 63.

Stanley, William. (1984). Economic Migrants or Refugees from Violence? A Time-Series Analysis of Salvadoran Migration to the United States. Latin American Research Review 22, 134. 144.

Trump, Donald. (2017). Remarks by President Trump to Law Enforcement Officials on MS-13. White House transcript of remarks. https://www.whitehouse.gov/the-press-office/2017/07/28/remarks-president-trump-law-enforcement-officials-ms-13.

U.S. Immigration and Customs Enforcement. (2017). Fiscal Year 2017 ICE Enforcement and Removal Operations Report. 4. https://www.ice.gov/sites/default/files/documents/Report/2017/iceEndOfYearFY2017.pdf.

Ward, T.W. (2013). Gangsters Without Borders: An Ethnography of a Salvadoran Street Gang. 3, 13.

Zamora, Javier. (2017). Unaccompanied. 9-10.

Zamora, Javier. Unaccompanied. Port Townsend: Copper Canyon Press, 2017.

Volume 11 Issue 1 | Spring 2018 RESEARCH

Just for laughs: Utilizing Machine Learning to Rate and Generate Humorous Analogies

Limor Gultchin
Harvard College '17

This thesis aims to present a procedure for generating humorous 4-word analogies, in the form of

humans : water :: Texans : barbecue.

By using a neural word embedding, we created a system that can construct 4-tuples of words of a comedic nature, based on an initial pool of funny analogies, written and rated by Amazon Mechanical Turk (AMT) users. Our procedure involved 4 main steps:

1. **Generating a collection of "funny words", by classifying with a support vector machine (SVM) words from the embedding that are similar to words used in the analogies written by AMT users.**
2. **Generating funny pairs of words taken from the collection of funny words, and classifying them to obtain more likely funny words. Negative examples were randomly generated pairs from the embedding, and positive examples were pairs from the AMT users' analogies.**
3. **Generating matchings of generated pairs by another SVM classifier. Negative examples were random 4-tuples of words from the embedding; positive examples were complete humorous analogies obtained from AMT.**

Our method was shown to perform significantly better than the following baselines:

- **random 4 tuples of words from the embedding**
- **random 4 tuples of words from the "funny pool" of words we classified**
- **random matching of funny pairs we generated.**

We assessed the performance in terms of the mean scores obtained per analogy in each baseline, and in terms of maximum funniness score obtained in each category (7/10 fully-computer generated, 5/10 random match of pairs, 4/10 random "funny" words and 3/10 random words). To further establish the usefulness of neural word embeddings to capture humor and generate comedic structures, we introduced a "funniness" score prediction which showed positive correlation with actual ratings obtained from AMT users, and performed a Turing test, in which 35% of computer-generated analogies were mistaken to be human made.

Introduction

Computational Humor: What's At Stake

With recent advance in machine learning and data science, more and more fields that were once sealed off from computational approaches are opening up for the touch of artificial intelligence. Humor is one such field. Notoriously difficult to analyze, assess and rate even for the human agent, it provides a challenge for the adventurous computer scientists who dare to take it on. Literature that provides analysis of the roots and causes of humor has been circulating at least since ancient Greece, with the following major theories presented to date[10]:

1. Superiority theory. The superiority theory of humor states that the feeling of selfcontainment associated with a sense of superiority in our status or well being over others is what makes us laugh. In other words, we laugh when we are better off than a fellow human, and our assertion of this fact. Notable philosophers proposed and bolstered this assumption, an act which did not help the status of comedians and humor: they were deemed as agents of evil mockery and pride. Among the famous supporters of this view were Plato, Descartes, Hobbes and the Bible.

2. Incongruity theory. Scholars endorsed a more modern and positive approach to humor was endorsed mostly since the renaissance. This reading of humor claims that what creates a comedic effect is a "benign violation of expectation"[8]. Recent research have shown that many types of laugh outbursts could be explained by their surprising nature. Whenever an unexpected event happened, that turned out to be harmless – amusement was afoot. Two conditions had to be met to create humor:

(a) an incongruity, which first took an audience by surprise, perhaps even a tense, fearful one.

(b) a resolution of the contradiction of expectation had to happen, followed by relief, which had a vocal manifestation, of the form of "Ha Ha", accompanied with an upward twitch of the lips.

Among the supporters of this understanding are James Beattie, Immanuel Kant, Arthur Schopenhauer and Søren Kierkegaard.

3. Stress release theory. Along the lines of the relief theme proposed by the "incongruists", Sigmond Freud proposed a similar reading into humor, with a typical Freudian twist. To Freud, humor was a result of pent up stresses, many of whom related to societally taboo subjects, such as sexuality and violence, that are released when a joke teller refers to them directly or as an innuendo. By "opening for a conversation" a subject the listener has been putting active effort to suppress, a joke can signify a promise of relief, an appeasing statement, an "it's OK" signal. It tells the listener others are contemplating and suppressing the same topics, and offers an opportunity to vent some of these accumulated suppressions. Lord Shaftesbury, Herbert Spencer and John Dewey also adopted a biological-psychological reading into humor as a stress reliever.

4. Mock-A ression play theory. From a biological-evolutionary point of view, laughter can be explained by its predecessors in primates. Observations by evolutionary biologists revealed that primates laugh too, usually during mock-aggression activity within

a family of chimpanzees. Mock-aggression activity can be seen when members of the same family of monkeys are engaged in playing, that to an outside observer might look like a violent confrontation. When two or more such primates are "fighting" in this way, like children who are engaged in a mock-fight, they seem to be training for actual potential future conflicts, yet need to signify to each other that they are not being truly aggressive. The chosen signals are usually a quick, fast breath of air coming from the diaphragm and a movement of the mouth, in which the front teeth are exposed and the lips are drawn back and upwards. The sound this mock-aggressive signal is making is "ah ah", which resembles human laughter, only in a backward direction: primates' diaphragm is set up a little differently than that of humans (or rather the other way around?), to allow for appropriate locomotion. Thus, primates breathe in instead of out, when poking fun at their fellow primates.

Computational Humor

A computational approach to humor is a much more recent development, yet it has already produced its own modest history. Beginning in the 1990s, computer scientists attempted at understanding, and moreover producing humor automatically. Developments in machine learning made it possible to imagine an algorithm that could take in examples of humor and to try uncover their inner pattern. HAHAcronym[14] was one of the first examples of the development of a humor-generation focused system, which aimed to produce humorous acronyms. In the mid 2000s, Twitter proved to be a useful platform for such investigations, due to its ease of access and its inclusion of ready-made initial classifications (e.g. hash-tags). Barbieri and Saggion utilized the popular social network to detect irony[3], while Yishay Raz focused on automatic classification of types of humor[12], such as anecdotes, fantasy, insult, irony, etc. Some of these computational approaches referenced the voluminous traditional thinking of humor, presented above[2]. There was also much work done for the understanding and assessment of visual humor. In 2015, Shahaf et al. joined Bob Mankoff, cartoon editor of the New Yorker, to build a system that would be able to predict which one of the +5,000 weekly submissions to the newspaper's cartoon caption competition is the funniest one[13]. These attempts and more have all been interesting and illuminating, but proved to have a limited amount of success. There is clearly much more to be done to achieve a more accurate understanding of what makes things funny and how we can "teach" humor to computers.

Increments to our knowledge of humor are gradual and modest, being a difficult task as it is, and yet there are more and more indications as to the potential of such investigations. In an attempt to add to the existing corpus of computational humor literature and experimentation, this thesis focuses on generating literal humor, through an examination of Google's word2vec word embedding. In particular, we will examine humorous 4-word analogies generated based on the embedding's representation of words.

Word Embeddings

For this literal approach to the composition of humorous 4-word analogies, we took advantage of the existing and relatively new technology of neural word embeddings, and in particular Google's word2vec, which opened to public usage in 2013119. Using a neural net trained on instances of Google news articles containing 3 billion running words, Google's machine learning engineers were able to create an embedding of 3 billion words mapped into vectors of 300 dimensions. The vectors are learned representations of the words in the training text corpus, which were crafted using continuous bag-of-words and skip-gram architectures. A neural net is trained to predict a word for the words that appear close by in the text, and the parameters learned by it are used as these word representations. In fact, a semantic field is created, such that words that tend to appear closer across the training texts appear closer to each other in this multidimensional space as well. The resulting vectors thus capture relations between words in the underlying training data, such as which words are similar to each other (and thus are 'neighbors' in the semantic space) and allow us to complete analogies that capture the relations between pairs of words (such as Paris:France::Rome:Italy). These resulting vectors can therefore be used as features when training models in various natural language processing and machine learning applications. For example, through word2vec, Bolukbasi et al. uncovered gender biases in the underlying embedding[6]. In this study, analogies such as

Man : Computer programmer :: Woman : homemaker

were automatically created after taking the cross product and Euclidean distance measures of vector representation of single words, pairs and quadruples. We decided to utilize this approach to study the humorous nature of association of words. If word embeddings can uncover gender biases, why can they not uncover funniness of words and combinations of phrases?

Linear regression and SVM classification

This paper had three main goals: to generate humorous analogies, to predict ratings of humorous analogies and to perform a Turing test to assess our results. In order to achieve them, it focuses on binary classification of words and analogies into "funny" and "not funny" categories, and on linear regression to generate prediction of "funniness" scores, based on ratings given by Amazon Mechanical Turk users. Following is a brief explanation of these two methods, meant to dispel the magical nature of machine learning as a "buzz word."

SVM classification

Once we had our data organized as numerical values, that represent features of different objects (provided by word2vec, in our case), we could train a classification model. Our approach required 4 different classifiers, which created somewhat of a "cascading" effect. SVM classifiers fit a linear classification separator between groups of data that have certain labels. In most tasks of binary classification, they are used to separate positive from negative examples. Our case was no different – we tried to separate positive funny examples from unfunny examples. The separator is fitted such that the least data points will be misclassified (having the opposite label than the desired one). SVMs, or support vector machines, are unique in their definition of a decision boundary which has a desired margin. In a classification problem treated with an SVM, we look at the points that are closest to the hyperplane as support vector (hence, the name). The certainty of a classification of a data point can be determined by its distance from the hyperplane (the farthest it is, the more certain the classification).[5] Thus, the best possible hyperplane we could fit is the one that maximizes the distance of the closest points (or SVs) to the hyperplane. A basic hyperplane can be defined as

$$h(x; w, w_0) = w^T x_i + w_0$$

The goal is to fit the classifier with lowest loss rate, where labels of the classification itself will be described as 1 if $h(x; w, w_0) > 0$ or -1 otherwise, and in our case as funny if $h(x; w, w_0) > 0$ or unfunny if $h(x; w, w_0) < 0$. When determining the weight vector w we will try to maximize the distance of the closest points, the support vectors, from the hyperplane, on both sides, while still maintaining a correct classification. To find the optimal hyperplane we will need to minimize the following expression:

$$\left[\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(w^T x_i + w_0))\right] + \lambda||w||^2$$

Linear Regression (Ridge)

The regression task was done with ridge regression, where a linear function was fitted to predict the scores (y axis) to match a data point, represented by the numerical value associated with a 4-tuple. The following is the definition of the regression[1]:

$$h(x, y) = Xw + w_0$$

And the loss function we aim to minimize is defined as:

$$L(w, w_0) = (y - Xw - w_0 1)^T (y - Xw - w_0 1) + \lambda w^T w$$

Where X is our features matrix (aka the featurized data), w is the weight chosen for the production of the regression line, w0 is the included bias terms and λ is the ridge regularization parameter. h is therefore the function for a prediction generation. The generation and rating process discussed in this thesis uses these two ideas from machine learning theory[5].

The implementation was made possible through the python library scikit-learn[7], which offers great support in putting machine learning theories into practice, and into actual predictions and classification models.

Contribution

In this project we furthered the understanding of humor and the capabilities of producing it "on demand". There are various benefits to the development of artificial humor capabilities:

1. Allowing the creation of smoother, more fun interfaces to use, which will surely play an even greater role in our lives in the coming years. Systems which include humorous components could be more congenial: making queries, tasks and warnings less repetitive, statements of ignorance more acceptable and error messages less patronizing[4].
2. Facilitating a better understanding of humor itself, by asserting or disproving notions of what makes things funny.
3. Overcoming yet another interesting artificial intelligence challenge posed to computer science researchers.

To the best of our knowledge, there has been no attempt to use our newly gained understanding of word embeddings in the field of computational humor. Furthermore, there has been limited success in previous attempts of humor generation tasks. This is yet another attempt at providing a proof of concept, for future research. This paper shows that computers can not only construct a humorous structure, but also recognize humorous themes relatively well, implying that it might have implications on a fuller understanding of what makes things funny.

Joking Around, or, Learning to Generate Funny Analogies

Our main goal was to show that humorous analogies can be generated based on the word embedding word2vec. We started from any random combination of 4 words, and tried to later generate funny 4-tuples. In the process, we trained 3 different classifiers, which match the 3 phases of generation:

1. 4 random words → 4 funny words. To start building our data set we asked Amazon Mechanical Turk users to come up with to 5 humorous analogies on any topic, and created an initial pool of about 1000 human written analogies. We then asked other users to rate those analogies, in the following manner: each participant was asked to indicate whether a batch of 25 analogies was funny or not (such that they could provide a single up-vote for each joke they found funny). Each batch of analogies was rated by 10 different participants, for a total of about 1200 analogies rated (1000 of them human written; around 200 were analogies we found funny, presented as a check, to make sure raters won't tire of repetitive analogies which might not be funny, and thus affect the quality of their rating). Since the score range of an analogy is between 0-10, and the average rating for analogy was around 2.5, we concluded an analogy had to gain 4 or more votes to be considered as funny. Next, we trained a classifier which treated as positive examples all the words that were used by AMT users in analogies they have written, and later were rated highly by their colleagues. The word embedding was used to obtain words representing the positive and negative examples, and to draw new words on which we fitted the classifier. Then, we could pull new words that were classified as funny, to create our collection of funny words.

   We decided to treat each of the words from funny-rated analogies as "funny" in itself for our initial classification, as we knew we needed a starting point for this demanding task. The negative examples were any word from the embedding, including verbs, generic names or propositions, which tend to be less likely to be part of a joke. We needed to create an initial collection of words that had a significant likelihood of appearing in a funny analogy. Thus, in a liberal yet effective manner, was treated all words already mentioned in funny analogies as having higher likelihood of appearing in jokes, and thus as generally "funny words".
2. 4 funny words → pairs of words. We made a new classifier to generate potentially funny pairs of words, from the pool of funny words. We trained another SVM model, and used the length of each word and the angle and distance between the vector representation of the words as features. After tuning the hyper-parameters of the model, we managed to classify quality pairs, using pairs from the Turkers analogies as positive examples, and random pairing of funny words as negative examples.
3. Generated pairs → classified matching of pairs. As a final stage in this cascading process, we trained a final SVM to tell the difference between random pairing of the 2-tuples and good pairings, which have an appropriate affinity between their first and second halves. The pairs we were using to create these full 4-word analogies were the generated pairs of the previous phase, and the features used were a combined 1200 dimension vector, made up of each words word2vec representation, as well as the distance and angle between the pairs. The Positive examples for training were,

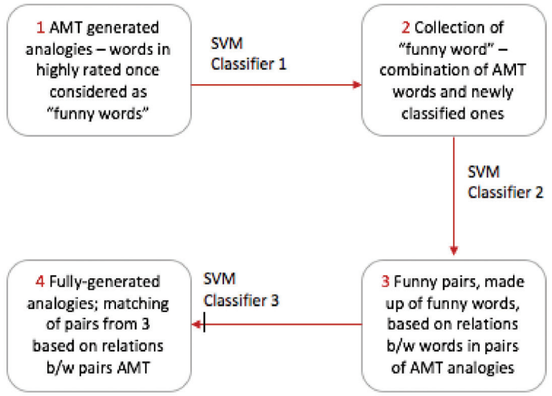

Figure 2.1: Genera on process schema

as expected, the full 4-tuple analogies rated funny among the AMT users, and the negative examples were random pairing of pairs generated in the last round.

Through this iterative yet evolving process, we managed to generate new analogies that could have now be put to the test of AMT users. The task that was presented to them was identical to the original rating task described above, with the sole difference that the analogies now presented were computer generated. We decided to use the following baselines (each consisting of 300 analogies tested) for our analysis, so that we could assess the progression of this method, one step at a time:

- 4-tuples of completely random 4 words from the embedding
- 4-tuples of funny words from the pool generated by classifier 1.
- 2-tuples of randomly matched generate pairs by classifier 2.
- AMT made analogies.

Results

Let us compare these 5 results (including the complete, newly generated analogies, made using classifier 3):

Comparing the top ranked jokes can also give us a sense of what the trend looks like. Among the different baselines, the most highly rated score achieved was a 5, for the analogy

woman : jungle :: hair : fathers

which belonged to the random pairs. Here are a few examples, with their corresponding scores, per each baseline:

1. Random-4 – the highest score was a 3, given to 10 analogies out of 300. Among them were

store shelves : shipyards :: NZX : Bowl Championship
renewed : knives :: policies : attempt
implementations : socializing :: overspending : fix.

2. Random-4-fun – the highest score was a 4, given to two analogies:

meal : metamphetamine :: disillusioned : Hendrix
bishop : appalled :: Australians : thermostat.

The first round of classification demonstrate a pretty good understanding of the kind of themes that drive humorous analogies, including references to pop culture, nationalities, religion, food and drugs. Other references that showed up in the pool were related to sex, family, animals (mostly dogs and cats) and sports.

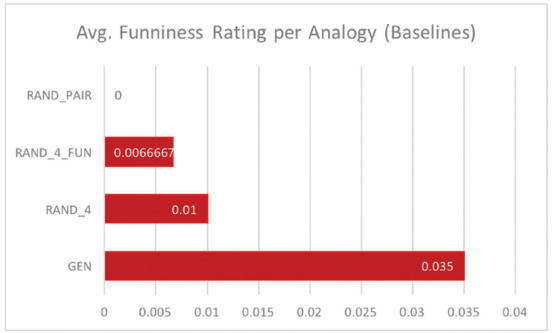

Figure 2.2: Average funniness scores for each baseline: random 4-tuples (rand-4), random 4-tuple of funny words (rand-4-fun), randomly matched classified pairs (rand-pairs) and fully generated pairs (GEN).

3. Random-pairs – the random pairs baseline results showed a single analogy that was rated at 5, and a couple that were rated as 4. The 5-rated was

woman : jungle :: hair : fathers

and the next two, rated at 4, were

stunt : governor :: petty : sex
ass : dems :: poodles : This.

Although most pairs seem promising, their combination is random, and therefore

requires further classification.

Now let us examine the case of the fully classified analogies (following the 3-step process described above). The highest ratings given were 7s and 6s: the most highly rated analogy was

water : Kardashians :: toilet : Reality TV

and a few 6ers were

bear : diamonds :: Trump : empathy
barking dog : McCain :: burglar : Vanilla Ice.

More results are summarized in Table 2.2.

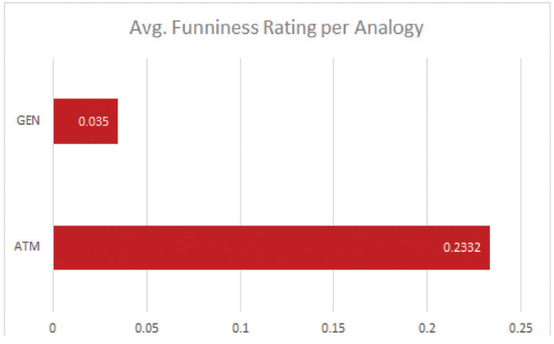

Figure 2.3: Average funniness scores for fully generated pairs (GEN) versus Average funniness scores for Amazon mechanical Turk users written analogies (AMT).

Rating vs. type	Random 4	Random-4-funny	Random-pairs	Full-generate
7	–	–	–	water : Kardashians :: toilet : Reality TV
6	–	–	–	Barking dog : McCain :: owner : Mitt Romney | George W Bush : unemployed :: Clinton : solitary confinement
5	–	–	Woman : jungle :: hair : fathers	Wife : music :: men : Gamecube | liberty : Donald Trump :: man : roulette | Obama : Selena Gomez :: Hillary : sedentary lifestyle
4	–	meal : methamphetamine :: disillusioned : Hendrix | bishop : appalled :: Australians : thermostat	stunt : governor :: petty : sex| ass : dems :: poodles : This	wife : adage :: grandmother : cliche | people : Gmail :: chicks : Facebook | successful : nerd :: high : doofus | billionaire : PCs :: Hillary Clinton : DOS | Trump : empathy :: luxury : welfare | PC : stupidity :: Apple : selfishness | Barack : Trade ::nukes : Trump
3	store shelves : shipyards :: NZX : Bowl Championship | renewed : knives :: policies : attempt | implementations: socializing:: overspending : fix	smoke detector : baking cookies :: Big : Raiders | shitty : socialization :: messed : gringo | pint : fattyfoods :: slang : MTV Movie Awards | pediatrician : boogeyman :: affordable housing :monotone	Watermelon : JK Rowling, Burger King : It| tofu : Republican, Walmart : Millionaire | testosterone : giggles:: Super Bowl : mess | student : geeks :: Charlie Sheen : corruption | sex : worship ::Republicans : cares	Women : procreation :: wives : sleeping | wife : advertising :: mother : buffoonery | vanity : maths :: Trump : twit | Twitter : Christmas :: Hitler : Santa | passion : millionaires :: disdain : money | homeowner : truthiness :: Trump : empathy | girls : fight :: men : comment | dumb : Blackberry :: cheesy : Facebook | cats : scammers :: bird : dangerous

Table 2.2: Breakdown of funniness ratings by type of baseline

Regression Towards a Laugh, or, How to Rate Funny Analogies

Predicting ratings AMT users would give to analogies (based on the same procedure that was described in the previous chapter) is another approach we could take to assess the types of humor understanding we can capture. Given an initial set of analogies and ratings provided by AMT users, we can define a regression line that would enable us to predict the scoring of a newly given analogy. The goal is to pick such a regression line y(x), such that the expected value of the loss will be minimized. For a more robust explanation of the training of such a model, please refer back to the section 1.4.2.

In our case, we started fitting the line with the data points representing the analogies given by AMT users, and using the ratings given by other AMT users as the data labels *t*. After tuning the weights in such a manner, we used the fitted regression line to predict the scores of new analogies generated by our system. The features for the regression were a 1200-dimension vector, the result of a concatenation of the word2vec 300-dimensions vector of each word. We chose ridge regression, which can lead to a sparse model that will be less likely to over-fit than a basic linear regression, and thus generalize better to new data points.

Our results showed positive correlation between our predictions and the scores which were eventually given by Turkers to our generated analogies (computer-generates ones). We performed a cross-validation, a technique which further helps us avoid over-fitting. To that aim, we separated the data to train and test sets (in our case, for a total of 5 groups) and could tune the hyper-parameters of the model without affecting the quality of our predictions, by modeling the noise in the training data so closely that it will negatively affect our accuracy for new data points. We chose a lambda value of 10 after performing a search for the value that will minimize the loss, and ran the regression. The results of this cross validation were giving a mean correlation score, between prediction and actual scores, of around 0.371 with a standard deviation of +/-0.038, for a 5-fold partition.

Though this positive correlation might be lower than we would aim for in typical regression, humor requires a different scale of assessment. Humor cannot be objectively agreed upon by different humans, let alone by a computational approach. Therefore, a positive correlation of 0.37 attests to an ability to predict scores of jokes in a satisfactory manner – one which performs much better than a random assignment of score predictions.

Comedy of Errors, or Applying the Turing Test to Our Jokes

Since the topic of this thesis is a machine learning question that is flirting with Artificial Intelligence, trying to make a machine "learn" a task that is deep within the realm of what humans define as intelligence. This chapter explores whether it is possible to fool humans into thinking some of the generated jokes were human

real/perceived	Human	Computer
Human	$\frac{28}{35} = 80\%$	$\frac{7}{35} = 20\%$
Computer	$\frac{12}{34} = 35.3\%$	$\frac{22}{34} = 64.7\%$

Table 4.1: Table of Confusion: Turing test full results

made (and vice versa). To achieve this goal, we created another Amazon Mechanical Turk task which consisted of 69 analogies – half were computer generated and half were human made. We chose the 35 highest rated human-made analogies and the 34 highest rated computer-made ones, to make sure that all analogies were in fact relatively successful jokes, which would make the turing test more focused on the humor aspect of this project, rather than the logical quality of the analogies. As a result, we could also keep the AMT users who participated in this test slightly more amused. We asked 10 participants to mark whether each of them was, to the best of their assessment, the work of a computer or a human. We then took a majority vote of the 10 users – each vote for "human" was considered as 1, and each vote for "computer" was considered as -1 (if no choice was made by a user, we considered it as a 0 vote). Then, if an overall score was positive the analogy was considered as human-made by the majority vote, and vice versa was considered as computer-made if the obtained score was negative. Mostly, AMT users were pretty successful at predicting the identity of the generator. However, out of 69 analogies, AMT users failed to classify correctly 19 of them, for an error rate of 27.5%. Overall, 12 computer-generated analogies were thought of as human-made by a majority vote of 10 users, and 7 human-made were mistaken for computer-generated ones.

Our generated analogies were mostly recognizable as computer-made. Yet, it provided evidence that this Turing test was far from trivial – out of 35 computer-made analogies, 12 were mistaken for a human-made analogy – a significant rate of 35%. The error rate of mistaking a human analogy to be a computer generated one was 20%, both quite surprising for a task that initially sounds rather intuitive.

Conclusion

In this thesis, we have shown how new word embedding techniques can be used for the enhancement and advancement of the study of computational humor, and humor at large. We have provided a description of a process for the production of humorous analogies, which proved to have been more successful than three different possible baselines. We were able to generate analogies which were rated as funny by AMT users, who did not know the analogies were produced by a computer. Furthermore, we have shown that there is a positive correlation between our predictions and scores AMT users will provide to analogies. Finally, we performed a Turing test, in which users were asked to identify whether certain analogies were made by humans or by a computer program. In doing so, users were unable to identify 35% of the computer generated analogies.

After showing a "proof of concept" in the form described above, this thesis further opens the door for enlarging the capabilities of this humor-generating system. Our current system seems to be doing a good job at recognizing themes and generals imageries which are favorable to humor creation. We can recognize it in the significant spike in the "funniness" of our analogies, past our random-4-funny-classifier, and by the milder improvements presented by the rest of the classifiers which are involved in our cascading humor generation system.

However, a lot more can be done to put together the successful pieces we identified. We can assert that marriage, family, sex, religion, politics and food are topics favorable to humor, and that verbs are less likely to be used by our classifiers. Yet, we can do more to combine the words in an analogy that truly captures a good joke. For example, using solely qualitative assessments, it seems that the fourth word can have a great effect on the cohesiveness of a humorous analogy in the same way that a punch-line is qualitatively assessed by stand-up comedians to "make or break" a typical joke. Therefore, here are a few further steps we would like to take in the future:

1. Train a "punchline" classifier, to produce better analogies, and once again assert, in a quantitative manner, a well-known qualitative notion in the world of comedy.
2. Train a classifier to recognize the optimal ordering of an analogy. For example, the analogy we rated in the first chapter

twitter : Christmas :: Hitler : Santa

could arguably sound better if written in a different ordering, i.e.

Christmas: Santa:: Twitter: Hitler.

We could train a classifier based on orderings of recognized funny analogies, using features such as the angle between the vectors representing each word, pairs, and all possible pairing of the 4 words participating in the analogy. In doing so, we should be able to capture this subtlety rather well, and produce even "tighter" analogies.

3. After the suggested improvements to the generation process described above we can introduce a new Turing test, similar to the one described in chapter 4, such that we can achieve an even better error rate (or, as we like to think of it, "rate of confusion"). Eventually, a competition against professional comedians could give us another measure of our system's performance.

Pursuing some of these future directions could produce funnier, tighter analogies, that will once again challenge our opinion on what computers can and cannot do.

Computer:humor::humans:humor?

Human-made mistaken for computer	Computer-made mistaken for human
Christmas : belly laughs :: Thanksgiving : depressing	Wife : nagging :: husband : slacking
Water : Kardashians :: toilet : Reality TV | Hillary : honest :: Trump : humble	PC : stupidity :: Apple : selfishness | Bear : gummy :: captain : crunch
Barack : Crazy :: Trump : aggro | Cat : burglar :: dog : friend	Wife : music :: men : Gamecube | Mom : no :: dad : Yes
George W. Bush : unemployed :: Clinton : solitary confinement	Woman : outlet :: man : plug
Barack : Trade :: nukes : Trump	David Copperfield : disappearing :: washing machine : socks

Table 4.2: Examples of misclassified jokes in an AMT Turing test, separated by kind of mistake.

References

[1] Alexander Rush, D. P. (2017). Linear regression and basis functions handouts. Harvard University, SEAS. https://github.com/harvard-ml-courses/cs181lectures/blob/master/03-lectures-lr.pdf.

[2] Bacciu, D., Gervasi, V., & Prencipe, G. (2016). Lol: An investigation into cybernetic humor, or: Can machines laugh? In LIPIcs-Leibniz International Proceedings in Informatics, volume 49: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[3] Barbieri, F. & Saggion, H. (2014). Modelling irony in twitter. In EACL (pp. 56–64).

[4] Binsted, K. et al. (1995). Using humour to make natural language interfaces more friendly. In Proceedings of the AI, ALife and Entertainment Workshop, Intern. Joint Conf. on Artificial Intelligence.

[5] Bishop, C. M. (2006). Pattern Recognition and Machine Learning (Information Science and Statistics). Secaucus, NJ, USA: Springer-Verlag New York, Inc.

[6] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In Advanc in Neural Information Processing Systems (pp. 4349–4357).

[7] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In ECML PKDD Workshop: Languag for Data Mining and Machine Learning (pp. 108–122).

[8] McGraw, A. P. & Warren, C. (2010). Benign violations: Making immoral behavior funny. Psychological Science, 21(8), 1141–1149.

[9] Miháltz, M. (2016). word2vec google news code implementation. github. https://github.com/mmihaltz/word2vec-GoogleNews-vectors.

[10] Morreall, J. (2016). Philosophy of humor. In E. N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2016 edition.

[11] Radim Rehurek, Tomas Mikolov, G. (2013). word2vec. Google Code Archive. https://code.google.com/archive/p/word2vec/.

[12] Raz, Y. (2012). Automatic humor classification on twitter. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologi : Student Research Workshop (pp. 66– 70).: Association for Computational Linguistics.

[13] Shahaf, D., Horvitz, E., & Mankoff, R. (2015). Inside jokes: Identifying humorous cartoon captions. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 1065–1074).: ACM.

[14] Stock, O. & Strapparava, C. (2003). Hahacronym: Humorous agents for humorous acronyms. Humor, 16(3), 297–314.