Joke Generation and Classification

Vasileios Moschopoulos, Georgios Michoulis and Dimitrios Tourgaidis

[1] Aristotle University of Thessaloniki, Department of Informatics, Data and Web Science Master's Programme, Thessaloniki, Greece

Keywords: Natural language processing, Machine Learning, Deep Learning, LSTM, Recurrent Networks, Text generation, Text classification

1 Introduction

Machine learning is one of the top tech topics in recent years, the companies widely use it, and it is extensively reported in the news [1]. Deep learning is one of the many machine learning fields, which is a trend nowadays because of the computer's evolution. Deep learning is essential to identifying faces and objects in digital pictures, analyze medical images [2], recognize the human voice in smartphones, and respond with search queries from the internet (e.g., Microsoft Cortana, Samsung Bixby). Soon this technology will help us with many other tasks. One of the uses of deep learning is to help machines understand the natural human language in order to generate text or classify it by following human language rules. Natural language processing is a subset of machine learning, which extends deep learning properties, with the purpose of making computers understand the natural human language concept.

Generating texts containing human sentiment is a hard task for deep learning. The linguistic properties of humor and its interpretation by human cognition is an open challenge. We believe training models with jokes in a supervised manner will give it more contextual data to infer from and generate creative content.

One of the recent and promising research domains is applying Recurrent Neural Networks (RNNs) on text models to prove the learning process. RNN models can learn text structures by training on a dataset at the input and then produce (generate) an acceptable (more or less) text in the output. The Long Short-Term Memory (LSTM) [3] model is a subset of the RRN architecture, with feedback connections that help better capture previous context in a continuous stream of text and is used in deep learning.

One of our goals for this work is to help an LSTM architecture network learn from sequences of jokes and generate joke-like text. We believe that by increasing the breadth of joke categories, a network can learn from various joke types and thus be able to produce cross-category and imaginative content, using the blend of information from training on multiple categorical data in a controlled manner. As a next step, we aim to classify the generated content based on its categories.

Text classification is the process of assigning tags or categories to text according to its content. In this paper, we use machine learning to create a classification model for the dataset we chose. We implemented two approaches, one using Naive Bayes and one using Logistic Regression. The two models classify the jokes into categories based on their content. Afterward, we use those models to classify the jokes generated from our LSTM-based model. Consequently, a further human effort will not be required to categorize them.

This paper is structured as follows. In the second section, we describe related work similar to our approach. In the third section, we describe i) the preprocessing procedure of our dataset, ii) the LSTM model architecture development and its challenges, and iii) the classification model development and the results of our experimentations with it. In the fourth section, we present our entire project pipeline and show examples of generated text classified into different joke classes. Finally, we conclude with a proposal for future work.

2 Related Work

Concerning joke generation, a case of unsupervised generation has been presented in [4] using 2-grams, where the form of the generated jokes is "like my X like I like my Y, Z", where X, Y, and Z are variables to be filled in. The jokes produced in [4] have a stricter structure compared to our generated jokes. On the other hand, more examples of supervised generation have been studied. A case of Japanese jokes generation is presented in [5], which are used sequentially for a stand-up comedy performance by robots. Then, a two-liner joke generator related to punning riddles [6] has been created, using the Semantic Script Theory of Humor. Lastly, using n-grams and lexical constraints, an example of 'Adult humor' generator [7] has been introduced.

As for text generation using RNN-LSTM networks, interesting research has been taking place until now. Initially, a story-line generator has been examined [8], which generates the stories based on a series of input stories. Then, a factual table generator has been introduced [6], where structured data has been used for the generation of the tables and their content, which is a set of field-value.

Finally, in the field of jokes classification, most work is related, initially to jokes recognition [9], whether a sentence is a joke or note, as well humor level detection [10], solely resolving the level of humor for jokes. We have taken a different approach related to the joke classification task compared to the above cases. We try to classify jokes to some predetermined categories, despite their level of humor, in which area not much research has been done.

3 Model

In this section, we introduce our approach to text generation, and consequently, classification. First, we describe our data preprocessing methods. Then we discuss text generation procedures and challenges we faced on building our LSTM-based generative model while presenting some generated joke samples. Finally, we describe our classification model development procedure and present our results.

3.1 Preprocessing

Preprocessing is an essential task before processing data for either text generation or text classification. Preprocessing is required before any experimentation, and it may affect the way in which outcomes of the final data processing can be interpreted. Data preprocessing is required because real data are missing attribute values or having only aggregate data, often they contain errors or outlier, and mainly, they are inconsistent. There are three main steps to achieve good preprocessing: i) data cleaning, ii) data integration and transformation, and iii) data reduction.

Firstly, data cleaning is also data scrubbing because it fills some missing values; it smooths noised from the date, and remove outliers. So, data cleaning is divided into four parts:

- Parsing, which locates and identifies individual data elements in the source files and then isolates these data elements in the target files (e.g., unique jokes or remove NaN fields).
- Correcting parsed individual data components using data algorithms (e.g., "can't" to “can not” and “-n't” to “not”.
- Standardizing in conversion routines to transform data into its preferred and consistent format using both standard and custom business rules (e.g., “ya ‘ll” to “you all”). Also, we set all characters to lower case.
- Searching and matching records within and across the parsed, and standardized data based on predefined rules to eliminate duplicated jokes.
- Consolidating by analyzing and identifying relationships between matched joke categories and merging them into one representation.

Finally, data integration and data reduction combine data from multiple sources into a coherent data store and obtain reduced representation in volume but produces similar analytical results. Moreover, we reduced our data by removing very small in sequence lengh jokes and we removed NaN rows and duplicated jokes.

We integrated joke categories in order to create better classification model to be able to classify jokes which were generated by our generation model. We combined the similar categories like “animal“ and “animals and we achieved better classification. “Animals”, “blonde jokes”, “insult”, “lawyer”, “light bulbs”, “medical”, “others” and, “redneck” are the final joke categories.

Our dataset comes from an online GitHub repository [11], which provided us three JSON files with jokes. Each file had different number of columns and rows, so, in order to integrate and combine two of the three dataset we had to reduce both datasets sizes.

To achieve the above actions, we had to choose proper python libraries and toolkits, such was the Regular expression operations (re library) and, the Natural Language Toolkit (NLTK) for the classification only. NLTK is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language [12]. We used this toolkit to normalize our dataset for example, we removed stop-words, we tokenized our dataset and, we lemmatized it to merge some tokens (e.g., rocks with rock). Re library was very helpful, because of this we were able to detect patterns and strings that we wanted to remove or transform from our dataset (e.g., “’ll” to “will” and remove double spaces). Further preprocessing applied on each part of our project separately and we will describe it in the following sub-sections.

3.2 Joke Generation

In this section, we present our approach and experiments on joke generation. This section is split into three main parts. We begin by describing how the dataset was loaded in a ready-to-input format for the model, we continue by presenting our model architecture and end describing five different approaches for joke generation. As a bonus part, we also describe an experiment that was made using an architecture which combines one-dimensional convolutional and LSTMs layers.

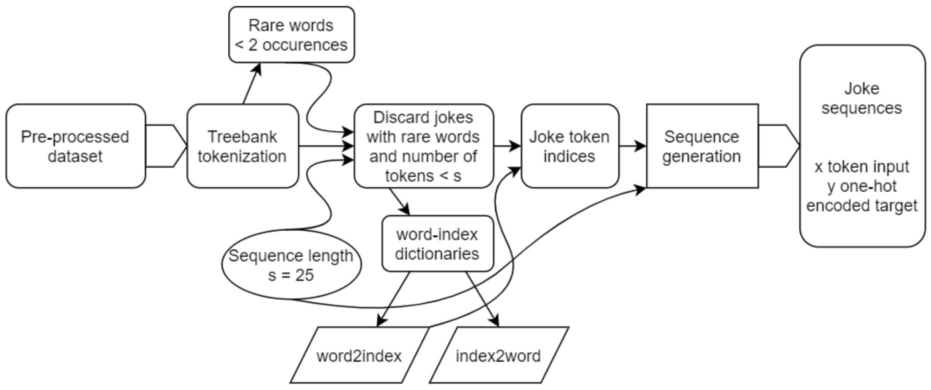

Fig. 1. Data preparation for model training

Data preparation For our data we used the preprocessed dataset, for which the procedure was described in section 3.1.

Jokes were tokenized using the Treebank word tokenizer and words having less than 2 occurences in the dataset were marked. Then, using a predetermined sequence length *s* ranging from 20 to 40, all jokes having less than *s* tokens and containing rare words were discarded. To produce realistic results, punctuation and stop words were kept. No lemmatization or stemming was performed.

Word-to-index and index-to-word dictionaries were created and then, using a scrollable sequence window over tokenized jokes, index sequences of size *s* were created. Based on the sequences, inputs and targets were created. Inputs are token indices x_i where i ranges from 1 to s - 1 in a sequence, while targets are the one-hot-encoded token indices y_s.

So, for sequences of s = 25 for example, x_i is the input containing token indices with i ranging from 1 to 24 and y_{25} is the one-hot-encoded token index in position 25.

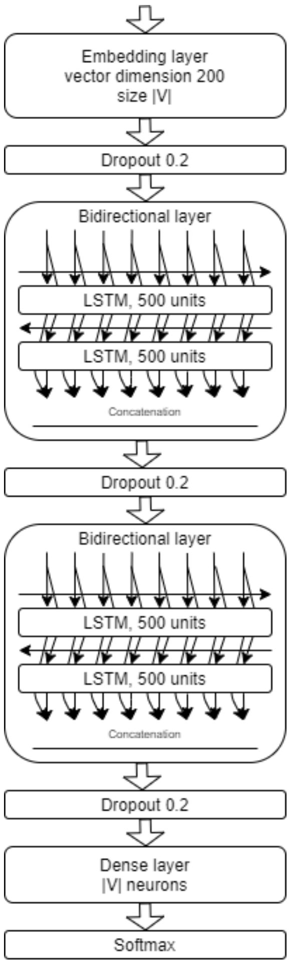

Fig. 2. Our bidirectional-LSTM-based generative model architecture

Model architecture Our model implementation is in Keras[1]. We created a 4-layer sequential network, with embeddings and bidirectional LSTMs [13].

At the first layer, an embedding layer was used with the size of our vocabulary and vector dimension of 200. Two bidirectional LSTMs of 500 units follow, using dropout of rate 0.2 in between. Finally, a dense linear layer having $|V|$ neurons, with $|V|$ being the size of our vocabulary, was used. Another dropout layer of 0.2 rate was used before the dense layer. A softmax activation function was appended to the end and the model was compiled using categorical cross-entropy loss.

The model was fitted for 23 epochs on a GPU with sequence length $s = 25$.

[1] Code source: https://github.com/Enkhai/joke-generation-nlp

Tables 1. Generated text samples using different methods. Number of generated tokens is set to 100 in all cases.
Seed text is `The man in the white suit tipped his hat. "Why do you keep looking at me like that?", he asked.`

Greedy decoding
the man in the white suit lowered his hat . `` why do you keep looking at me like that ? `` , he asked . `` i am , " said the man , `` it is just that i have changed you , and i have no good and some saying that you have to worry about them . " `` well , " said the doctor , `` i am going to see you . " `` i am going to be the same way , " says the doctor . `` i am the only store that i am going to be an army agent ! " `` well , " the doctor said , `` i am going to see you

Beam search, k=8
the man in the white suit lowered his hat . `` why do you keep looking at me like that ? `` , he asked . `` well , " he said , `` i am going to see you . " `` well , " said the man , `` but i do not know what i am doing . " `` well , " said the farmer , `` i am going to see you . " `` but i am not going to tell you , " replied the farmer . `` well , i will tell you , " said the blonde , `` it is a bet . " so the man asked her , `` what do you know ?

Sampling, temperature=0.4
the man in the white suit lowered his hat . `` why do you keep looking at me like that ? `` , he asked . `` i am , " she replied . `` i am just saying that the porch is to care , " he replied . `` well , " he said , `` your father is really late . " he walked up to her and said , `` you want to be a plane , it is your daughter has ever seen . " the woman replied , `` well , you are a big man , but we have a look on our wall . " the bartender says , `` you are not going to be a new

Top-k sampling, temperature=0.4, k=5
the man in the white suit lowered his hat . `` why do you keep looking at me like that ? `` , he asked . `` i am going to talk about , " she said , `` and i am getting pretty good . " the man said , `` well , i am not a bit my wife to find you . " the third man says , `` well , you have been getting some more than a moment . " the man says , `` oh no , it is not , we have all be a thousand dollars . " the teacher says , `` well , you have a kid , will not you ? " the man replies

Top-p sampling, temperature=0.4, p=0.005
the man in the white suit lowered his hat . `` why do you keep looking at me like that ? `` , he asked . `` i have you , " he replied , `` i am just saying that you can not read the bathroom . " the teacher said , `` well , i am a same thing , i was a guy ! " the japanese looks say , `` i am here , i am a man . " the bartender says , `` why ? " the man explained , `` i am looking for a couple of times because i ask you a problem . " the husband said , `` well , you have been pregnant , and

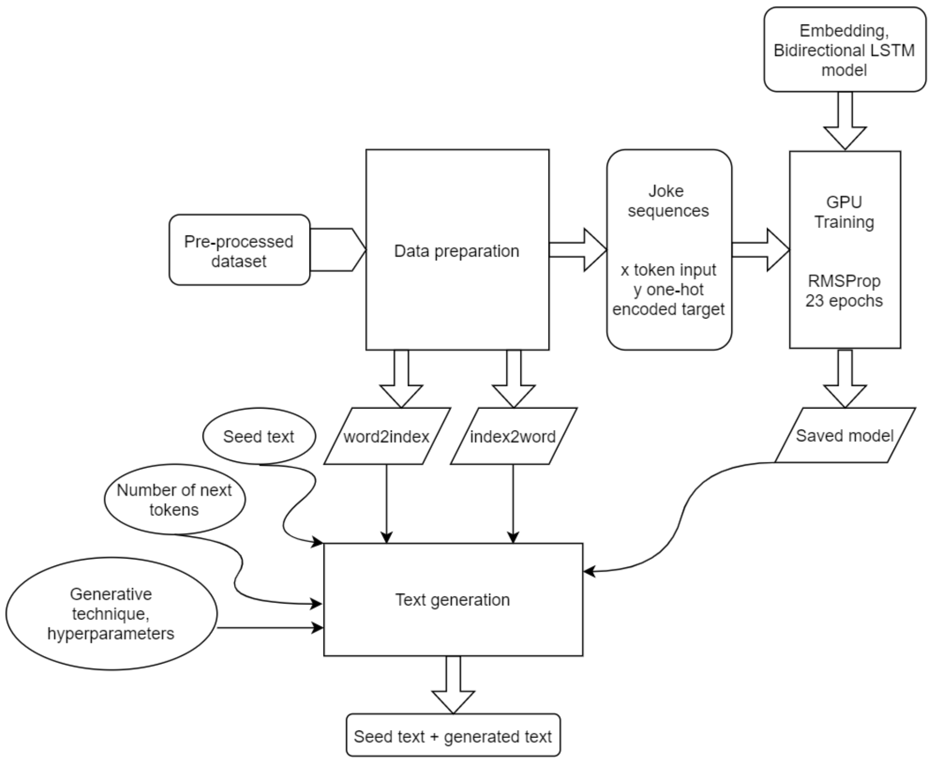

Fig. 3. The entire text generation flow

Text generation After the model was trained, we saved it in a file along with the word-to-index and index-to-word dictionaries to be used for inference, or in our case, generation.

Generation begins with a seed text, preferably having as many tokens as the sequence length the model was trained with. The seed text is then cast into lower case, tokenized using the Treebank tokenizer and transformed into token indices using the word-to-index dictionary.

For generation, we implemented five different approaches:

- Greedy search decoding picks the next most probable token
- Beam search decoding picks the topmost likely sequences based on a beam length parameter
- Sampling performs a simple sampling technique, picking the next token randomly using the model's token probability distribution prediction and temperature probability scaling. Temperature values less than 1 increase the probability of more probable next tokens, while values larger than 1 make the next-token probability distribution smoother.
- Top-k sampling performs sampling, filtering the top-k most probable tokens [14]

- Top-p sampling performs sampling, filtering the tokens having a probability higher than a specified value [15]

We also use a specified number of next tokens to be predicted. When the predicted number of tokens has been fulfilled, generation stops, and the entire index sequence is returned to be translated into text using the index-to-word dictionary.

We also experimented with an EOS, SOS and PAD tokens implementation using sequences of maximum joke token length, with pre-padding and n-grams of increasingly higher order. In this experiment text generation would end with an EOS token. The experiment produced poor results and was abandoned.

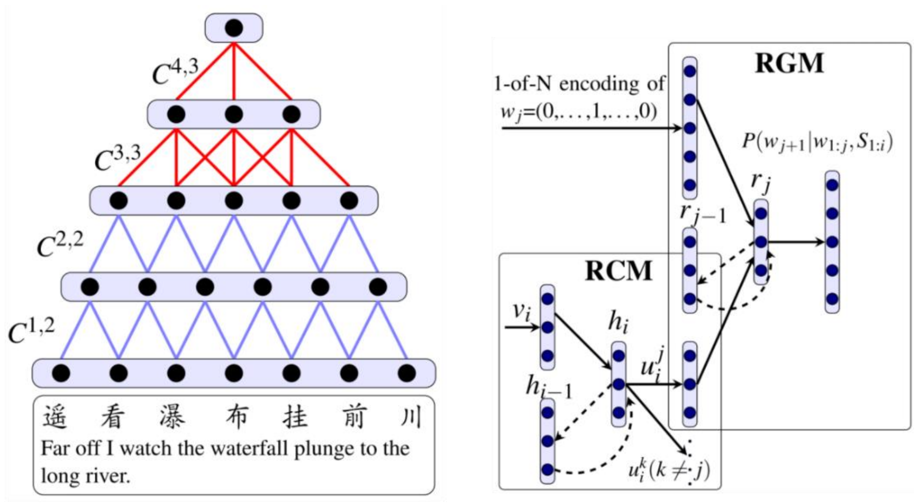
$C^{4,3}$
$C^{3,3}$
$C^{2,2}$
$C^{1,2}$
遥 看 瀑 布 挂 前 川
Far off I watch the waterfall plunge to the long river.
RGM
1-of-N encoding of w_j=(0,...,1,...,0)
$P(w_{j+1}|w_{1:j}, S_{1:i})$
r_j
r_{j-1}
RCM
v_i
h_i
h_{i-1}
u_i^j
$u_i^k(k \neq j)$

Fig. 4. Convolutional and recurrent network layers architecture presented in [16]

Experimentation with 1-dimensional CNNs and LSTMs As a bonus, we also experimented with the architecture presented in [16], which we implemented using LSTMs instead of RNNs.

The architecture, originally designed to produce Chinese poetry, is constructed by a convolutional and two recurrent networks as such:

- **Convolutional Sentence Model (CSM)**: Each symbol in the vocabulary corresponds to an embedding vector of dimension 200. Sentences of 7 symbols are passed through the embedding layer and then through a 4-layer 1-dimensional convolutional network. The kernel size is 2 in the first two layers and 3 in the last two for the convolutional network. As a result, each sentence is eventually represented by a vector *v* of dimension 200.
- **Recurrent Context Model (RCM)**: The sentence representation vector *v* is then passed through a recurrent layer to produce the context vector *u*.
- **Recurrent Generation Model (RGM)**: The last symbol in the sentence is one-hot-encoded, multiplied by a matrix *X*, and passed on through a second recurrent layer, along with the context vector *u*, to predict the next symbol

with a final linear dense layer, having neurons as many as symbols in the dictionary.

For dataset preprocessing we used a sequence length of 20, with each element in the sequence containing seven tokens to be processed by the convolutional layers. The previous words inputs were the one-hot-encoded last tokens of each element in the sequence, while the target token was the very last token in the sequence in one-hot-encoded format.

So, for a sequence length of 20, we would scroll through the jokes 27 tokens at a time. The first 26 tokens were used to build each element of the sequence, containing 7 tokens, and the previous tokens, while the 27th was the target token. The experiment ended with poor results and was abandoned.

3.3 Joke Classification

In this subsection, we present how we handled the text classification task. Generally, our objective was to create a model which can classify the jokes into as many as possible categories and simultaneously have good metrics. The classification task we tried to solve has two key characteristics. At first, each category has some key words, which are indicating the category in which a joke should be classified. Secondly, the concurrency of those key words is irrelevant to which category a joke should be classified. We decided to use the tf-idf model, in order to train our machine learning models, due to the fact that the characteristics of the tf-idf model are identical to the two characteristics of the classification task we try to solve. TF-IDF concerns about the importance of the words in a document, but doesn't concerns about the sequence of the words in a document, as well the words that are placed before or after a word.

We trained 3 different machine learning models; multiclass logistic regression, random forest and multinomial Naïve Bays, all of them implemented through python's sklearn library. For the training purpose of our 3 machine learning models, at first, we separated our dataset into training data and testing data, where the testing data was constituted by the 20% of all the jokes and the training data was constituted by the rest, solely the 80%. After the creation of the training and testing datasets, we created a matrix of token counts based on the jokes of the training dataset. For the purpose of the matrix token counts creation, at first, we created a vocabulary that contains 1500 tokens extracted from the corpus, solely the jokes, based on their frequency. Then we excluded words that exist in the 85% of the jokes, as well exist in an English stop word library of python's nltk library. Afterwards, we transformed the token count matrix into a normalized TF-IDF representation and then we continued with the training of our 3 distinct models. At first, we enabled our logistic regression model to be able to perform multiclass classification instead of binary classification, which it does by default. Furthermore, we used the liblinear algorithm to solve the optimization problem. Then, related to our random forest model, we set the number of trees to 50. The 3 tables below show the metrics of the 3 models, where the logistic regression model has the best metrics in comparison to the other 2 models, consequently it has been selected in order to classify the generated jokes in our project.

Table 2. The metrics of the logistic regression model.

Categories/ Metrics	Precision	Recall	F1-Score
Animals	0.52	0.42	0.47
Blonde jokes	0.88	0.77	0.82
Insult	0.36	0.17	0.24
Lawyer	0.94	0.60	0.73
Light bulbs	0.60	0.95	0.73
Medical	0.63	0.44	0.52
Others	0.87	0.93	0.90
Redneck	0.93	0.73	0.82
Accuracy			0.83
Macro avg	0.72	0.63	0.65
Weighted avg	0.82	0.83	0.82

Table 3. The metrics of the random forest model.

Categories/ Metrics	Precision	Recall	F1-Score
Animals	0.33	0.45	0.38
Blonde jokes	0.75	0.78	0.76
Insult	0.13	0.24	0.17
Lawyer	0.31	0.60	0.41
Light bulbs	0.72	0.95	0.82
Medical	0.22	0.37	0.27
Others	0.85	0.75	0.80
Redneck	0.71	0.58	0.64
Accuracy			0.70
Macro avg	0.50	0.59	0.53
Weighted avg	0.75	0.70	0.72

Table 4. The metrics of the multinomial Naïve Bayes model.

Categories/ Metrics	Precision	Recall	F1-Score
Animals	0.57	0.30	0.39
Blonde jokes	0.95	0.74	0.83
Insult	0.11	0.01	0.03
Lawyer	0.78	0.28	0.41
Light bulbs	0.64	0.85	0.77
Medical	0.62	0.19	0.29
Others	0.83	0.96	0.89
Redneck	0.94	0.63	0.76
Accuracy			0.82
Macro avg	0.68	0.51	0.55
Weighted avg	0.79	0.82	0.79

4 Project's Workflow

In this section we present how all the aforementioned tasks, analyzed in section 3, are combined together in order to fulfil the objective of our project, which is to generate jokes and categorize them based on their content and finally archive them in a file. Figure 5 presents the workflow of our approach.

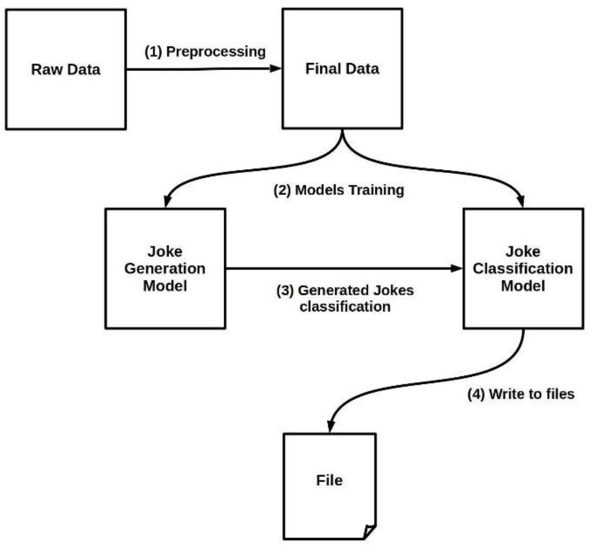

Fig. 5. Workflow of the project

Analyzing Figure 3 from top to bottom, at stage (1), the raw jokes are processed, resulting to the final dataset which is used a stage (2) to train the joke generation and joke classification models. Then, at stage (3), the jokes generated from the joke generation model are directed to the joke classification model, where at stage (4) the joke classification model categorized the generated jokes based on their content and archive them in a file with their categories. Selective results of our project are presented, where the first presented joke has been classified as ‘Blond’ joke and the second as a “Redneck” joke.

Seed text: ‘A blonde goes to the library and says to the librarian ’
Classified as: Blonde
A blonde goes to the library and says to the librarian. the man says, `` I would give you a drink, so i am going to take you on my husband. " the third says, `` well, I am a blond, i am going to take any more. " the second one said, `` I am sorry, but I am going to make my own new elephant with my wife. " the bartender says, `` well, you should have been any on the world? " the blonde replied, `` well, I am a mistake

Seed text: ‘A parrot, a horse and a lion walk into a bar. The parrot says ‘
Classsified as: Redneck
a parrot, a horse and a lion walk into a bar. the parrot says, `` you are the best thing I have ever had to be gone. " the other one says, `` I am going to be an ice cream restaurant, but it is the same thing i have ever seen. " `` but, "said the farmer, `` we are too old to be on my heart!" `` yes, "says the farmer, `` I am going to be any more that I am getting a new, and I must have to live on the house. " `` well

5 Conclusion

Soon an AI service may write, or at least draft, the necessary texts, which today still require human effort. Some of its most significant consequences are already imaginable. New models of deep learning are being developed at a growing pace and all the novelties are often difficult to keep track of. That said, for common natural language processing tasks, one specific neural network model has proven to be particularly successful. The model is called a Transformer.

Transformers were introduced by Vaswani et al. [17] in 2017 and were designed around the concept of the attention mechanism which was designed to help memorize long source sentences in neural machine translation. Transformers have some advandances compared to LSTMs. The main charecteristics are:

- **Non sequential**: sentences are processed as a whole rather than word by word.
- **Self Attention**: this is the newly introduced 'unit' used to compute similarity scores between words in a sentence.
- **Positional embeddings**: another innovation introduced to replace recurrence. The idea is to use fixed or learned weights which encode information related to a specific position of a token in a sentence.

GPT-3 is a transformer technology and a very promising one. GPT-3 writes a text continuing the sequence of our words (the prompt), without any understanding. And it keeps doing so, for the length of the text specified, no matter whether the task in itself is easy or difficult, reasonable or unreasonable, meaningful or meaningless. GPT-3 produces the text that is a statistically good fit, given the starting text, without supervision, input or training concerning the “right” or “correct” or “true” text that should follow the prompt.

As a future work we would like to try using GPT-3 or even better to implement and expand this algorithm, adapting it to our dataset. We strongly believe that this technology will produce more human like jokes and then we would like to use a humor sentiment analysis (which is also very new topic [18]), to be possible to fully automate “funny” joke generation, and would be a big breakthrough technology.

When it comes to text generation, an increase in the total number of jokes can also help reduce the number of epochs needed for the model to learn, as well as increase text diversity This, however, requires a much larger dataset that will allow us to trim without considerably reducing the amount of text available. As a bonus, top-k sampling can also be combined with top-p sampling for future experimentation.

For future work related to the classification task, we have in mind to experiment further with the preprocessing of the dataset, for example use different stop words libraries, as well to experiment, what results our model can produce for different number of categories, especially in the case where of finding more labeled jokes, which is the main obstacle in this effort.

References

1. Vigo, J.: Why Machine Learning Is The Future Of Business Culture, https://www.forbes.com/sites/julianvigo/2019/06/06/why-machine-learning-is-the-future-of-business-culture/, last accessed 2021/01/01.
2. Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., van der Laak, J.A.W.M., van Ginneken, B., Sánchez, C.I.: A survey on deep learning in medical image analysis. Medical Image Analysis. 42, 60–88 (2017). https://doi.org/10.1016/j.media.2017.07.005.
3. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to Forget: Continual Prediction with LSTM. Neural Computation. 12, 2451–2471 (2000). https://doi.org/10.1162/089976600300015015.
4. Petrović, S., Matthews, D.: Unsupervised joke generation from big data. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 228–232. Association for Computational Linguistics, Sofia, Bulgaria (2013).
5. Sjöbergh, J., Araki, K.: A Complete and Modestly Funny System for Generating and Performing Japanese Stand-Up Comedy. In: Coling 2008: Companion volume: Posters. pp. 111–114. Coling 2008 Organizing Committee, Manchester, UK (2008).
6. Labutov, I., Lipson, H.: Humor as Circuits in Semantic Networks. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 150–155. Association for Computational Linguistics, Jeju Island, Korea (2012).
7. Valitutti, A., Toivonen, H., Doucet, A., Toivanen, J.M.: “Let Everything Turn Well in Your Wife”: Generation of Adult Humor Using Lexical Constraints. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 243–248. Association for Computational Linguistics, Sofia, Bulgaria (2013).
8. Assistant Professor, K. J. Somaiya College of Engineering, Department of IT, Mumbai, Pawade, D., Sakhapara, A., Jain, M., Jain, N., Gada, K.: Story Scrambler - Automatic Text Generation Using Word Level RNN-LSTM. IJITCS. 10, 44–53 (2018). https://doi.org/10.5815/ijitcs.2018.06.05.
9. Mihalcea, R., Strapparava, C.: Making Computers Laugh: Investigations in Automatic Humor Recognition. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing. pp. 531–538. Association for Computational Linguistics, Vancouver, British Columbia, Canada (2005).
10. Weller, O., Seppi, K.: Humor Detection: A Transformer Gets the Last Laugh. arXiv:1909.00252 [cs]. (2019).

11. Pungas, T.: A dataset of English plaintext jokes. (2020).
12. Natural Language Toolkit, https://en.wikipedia.org/w/index.php?title=Natural_Language_Toolkit&oldid=993426778, (2020).
13. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. 45, 2673–2681 (1997). https://doi.org/10.1109/78.650093.
14. Fan, A., Lewis, M., Dauphin, Y.: Hierarchical Neural Story Generation. arXiv:1805.04833 [cs]. (2018).
15. Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y.: The Curious Case of Neural Text Degeneration. arXiv:1904.09751 [cs]. (2020).
16. Zhang, X., Lapata, M.: Chinese Poetry Generation with Recurrent Neural Networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 670–680. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/D14-1074.
17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. arXiv:1706.03762 [cs]. (2017).
18. Li, D., Rzepka, R., Ptaszynski, M., Araki, K.: HEMOS: A novel deep learning-based fine-grained humor detecting method for sentiment analysis of social media. Information Processing & Management. 57, 102290 (2020). https://doi.org/10.1016/j.ipm.2020.102290.