S-ColBERT vs. ColBERT: An Investigation of Simplified vs. Complex Models for Humor Detection

Yousif Alsaffar, Majd Barghouti, Lorenz Erlich, Saad Sahir, Dyanna Rivera, Youssef Ahmed

IE University

Natural Language Processing

25 April 2023

Table of Contents

Abstract

This paper investigates in depth, humour detection in NLP systems. Our study focuses on the linguistics and humour theories, with a particular emphasis on the Semantic Script Theory of Humor (SSTH) and incongruity theory. In addition, we examine the performance of modern architectures and their benchmarks such as ColBERT, XGBoost, and XLNet in recognizing humour. Using the knowledge obtained from these models/benchmarks, we want to expand our understanding of humour detection in NLP by comparing the strengths and drawbacks of various models, paving the way for better systems that can comprehend semantic information and contextual details better.

Introduction

Humor plays a crucial role in human communication, promoting social bonding and emotional expression. Building an artificial system to detect such a complex concept is a still an ongoing challenge to this day, largely due to the intricate and context-dependent nature of jokes. Our research begins with a thorough examination of the area of linguistics, with an emphasis on two fundamental humour theories, the Semantic Script Theory of Humor (SSTH) and incongruity theory. These theories provide a strong framework for linguistically defining humour and provide significant insights into the basic aspects that make a piece of text funny.

Following that, we look at the performance and approach of many cutting-edge architectures for detecting humor, such as ColBERT, XGBoost, and XLNet. Our research aims to emphasize the advantages and disadvantages of each model, as well as the importance of contextual information in the comedy detection process. By comparing these models, we expect to discover the most effective methodologies and approaches for detecting humor in language-based systems. Furthermore, we hope to discover possible areas for improvement and innovation in humor detection algorithms, ultimately leading to the creation of more complex systems capable of effectively grasping semantic information and contextual details in humor detection.

Literature Review

Humor Structure in Linguistics

Since we are looking at a language-based system, it makes sense that part of our research involved investigation into the field of linguistics. In the case of detecting humour, the central question is - what constitutes humour, linguistically speaking? Considering that a machine is unable to read into emotional nuances based on raw speech alone, we must understand what objective factors make a piece of text ‘funny’. In particular, during our investigation we found that the ColBERT paper [1] is centred around the Semantic Script Theory of Humor (SSTH). This theory states that a text must have two related but distinct scripts which are opposite to each other in nature. A script will contain large chunks of semantic information surrounding any given word and a cognitive structure evoked by it that is internalised by a native speaker. In the context of a computer algorithm, we will not have a native speaker internalising the context, but rather, an algorithm that learns based on this semantic information.

SSTH also links us over to incongruity theory in the field of linguistics [4] which states that humour is produced by an identified incongruity and the resolution of said incongruity. Putting the two together, we can conclude then that humour can linguistically be defined as two distinct scripts which contradict each other, but in which the contradiction is resolved by the end of the scripts (informally known as the punchline of a joke).

State-of-the-art Architectures

Taking into account the linguistic theory we just discussed, let’s now look at some of the previous state-of-the-art architectures. The ColBERT model, which outperformed other alternatives on the list, involves a single-path neural network structure to view the text as a whole, and then several other paths to view each sentence separately [1]. The sentences are then encoded numerically using BERT sentence embedding. This integrates the critical context that was mentioned as part of the SSTH theory in comprehending text as humorous, doing so in a way that is readable for a computer system by looking at each individual sentence and quantifying its values.

Meanwhile, XGBoost integrates regression trees into its learning process. In each regression tree within the architecture, each leaf has a continuous score which is represented as a weight and then classified based on decision rules [2]. However unlike ColBERT, XGBoost fails to view the text as a whole, only performing partial evaluation on each segment of it. XLNet is similar to XGBoost in this sense, as it only performs partial prediction as well based on a subset of text.

However, in the case of XLNet, the context is detected by looking at the position of a word and encoding both the current state and the surrounding context (in a similar way as attention-based transformers do with hidden states) [5]. The representation of the query (current state) only holds purely contextual information, unlike ColBERT, which also takes the actual content of a query into account. Nonetheless, it is evident how important such contextual information is when compared to the importance of the content of individual subsegments as we can see based on the figure below that both ColBERT and XLNet perform significantly better than XGBoost at detecting humour.

Previous Models' Findings [3]

Method	Configuration	Accuracy	Precision	Recall	F1
Decision Tree	N/A	0.786	0.769	0.821	0.794
SVM	sigmoid, gamma=1.0	0.872	0.869	0.88	0.874
Multinomial NB	alpha=0.2	0.876	0.863	0.902	0.882
XGBoost	N/A	0.72	0.753	0.777	0.813
XLNet	XLNet-Large-Cased	0.916	0.872	0.973	0.92
COLBERT	**N/A**	**0.982**	**0.99**	**0.974**	**0.982**

Data Pre-Processing

The selected dataset to train and test the model is a publicly available kaggle csv file containing 200,000 short texts for humour detection [7]. The dataset was made available as part of a research paper on humour detection utilising Google's BERT sentence embedding. Though the dataset in raw form is relatively clean there are various steps which need to be taken in order to construct valid input for the final model.

Dropping duplicate rows is part of general data cleaning to make sure the input is unique and not repeated. A filter was applied on the dataset to isolate the texts to contain only rows which were suitable for the application of humour detection. First the words and characters in each short text were counted, the dataset was then cut down to texts containing between 30 to 100 characters as well as texts with 5 to 25 words. This was mainly for two reasons, those being that texts which contain fewer than 5 words or 30 characters tend to be too short to consistently evaluate in an accurate manner. Humour typically relies on setting up a punchline and executing on it, which can be difficult given a text which is too short. Additionally an upper bound of 100 characters and 25 words was set partly due to computational reasons as well as ease of feature extraction. It can be difficult to generate representative embeddings for longer texts both in terms of accurate vector representation as well as the computational power needed to generate the embeddings. Additionally, the proposed architecture of the model suggests each short text needs to be split into 2 sentences, with separate embedded vectors for each sentence. For texts containing more words than our upper bound or fewer words than our lower bound it tends to be much more difficult to gain consistent and reflective vector representations.

To get the embeddings for each text after splitting each string into 2 distinct sentences the Bert-small-uncased model and its BertTokenizer was used [6]. The model is pretrained and freely available for use on Huggingface. For simplicity purposes the embedding was done in batches to ensure no memory failure, the batch size being 8 texts at a given time. Each text was tokenized and the Bert model was applied to get the respective embeddings of each sentence. The embeddings were then saved to a separate list for further usage. When tokenizing, padding and truncation was set to True in order to ensure correct input and output after transforming.

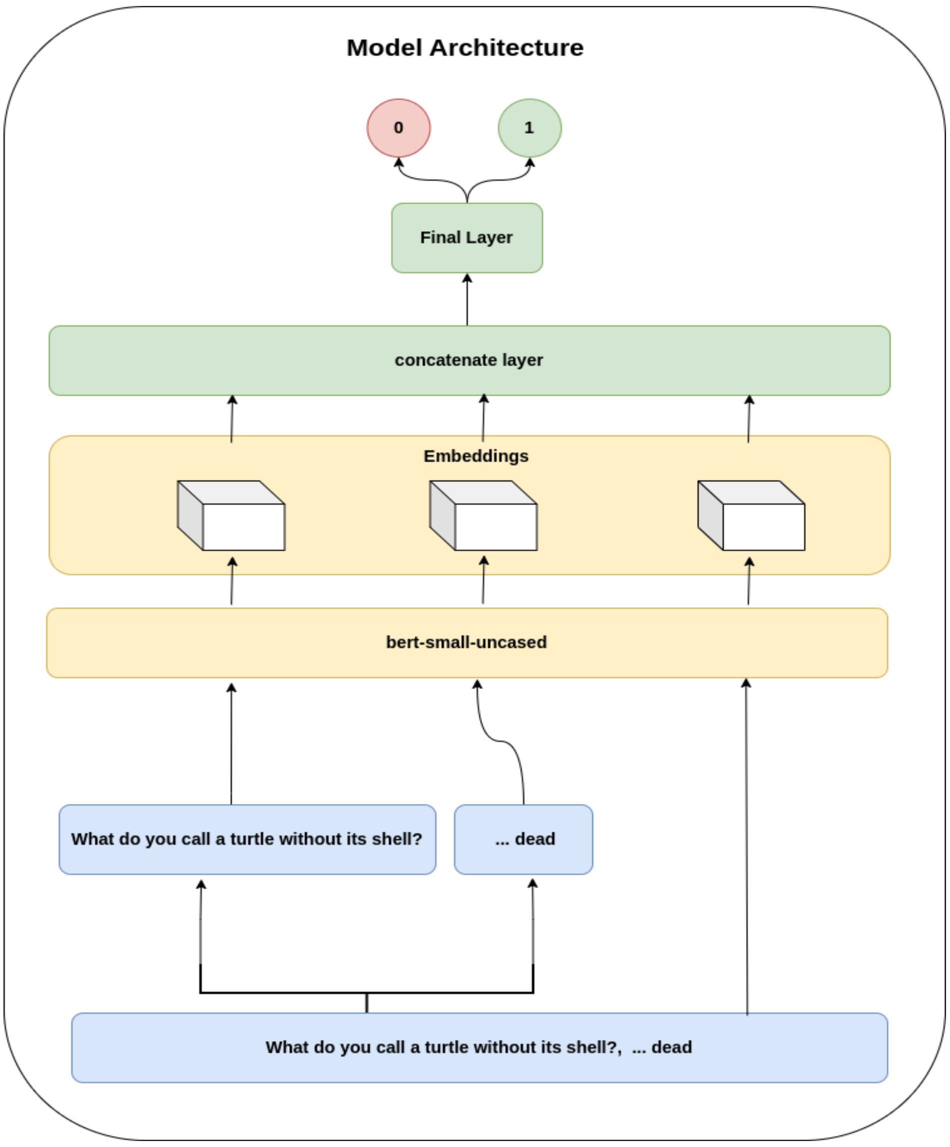
Model Development

Quantifying the general structure of a joke is essential to understanding the model architecture as analysing this structure helps understand the fundamental features that categorise the text as humorous. There are many sources in the study of the functioning of humour that define jokes as the sudden transformation of the perciever's expectations into nothing then nullifying that initial expectation by bringing humour to its lack of congruency. Based on this principle, the structure of humour can be classified into stages of storytelling that finally concludes with a punchline. Our model follows the Semantic Script Theory of Humor (SSTH) [4] which states that a text has to have two distinct related scripts that are opposites in definition. Based on this textbook definition of the structure of humour, if the model iterates through each sentence in the joke separately, each sentence will be labelled as normal and not funny. So the approach to follow is to make sure the model comprehends every sentence in the joke together under one defined context, and prevents any linguistic confusions on the model's part. The proposed method for detecting the structure of humour is based on the linguistic characteristic that when sentences of a joke are read separately, they appear normal and non-humorous, but when comprehended together in one context or line of the story, they become humorous.

The model defines relevant features as multiple hidden layers with specific purposes. The training of the model has to be done with both the split sentence vector embeddings as well as the whole text vector embeddings. This approach has shown to be of significant value for context localisation within the text [1]. Inputting the split sentence representation can allow for much more refined decisions about the context of each sentence to be made. This helps the model to focus on the context of the humour elements within different parts of the text, such as puns and incongruity. The processing of whole text vector embeddings allows the model to also learn the syntax of humour in the scope of sentence relationships, such as build ups and punchlines. Combining both of these methods ensures that contextual information is extracted at both a local and global level and thus helps to provide a deeper comprehension of what defines humour in a text. This can make the final model significantly more robust to different scenarios of jokes, whether they depend on contextual puns or a punchline.

Hidden layers within the neural network are used on the whole text as well as the split sentences, the split sentences are then concatenated together to recreate the full text, the whole text is then concatenated with the recreated text to capture as much information as possible. Following these operations a multitude of hidden layers are applied on the combined embeddings until a final output layer returns a decision on whether the text should be classified as humour or not.

Model Architecture


Model Architecture
0
1
Final Layer
concatenate layer
Embeddings
bert-small-uncased
What do you call a turtle without its shell?
... dead
What do you call a turtle without its shell?, ... dead

The architecture of the proposed method contains separate paths of hidden layers designed to extract latent features from each sentence and a separate path to extract latent features of the whole text. The sentence embedding for each sentence is fed into parallel hidden layers of the neural network to extract mid-level features related to context, type of sentence, etc. While the main idea is to detect existing relationships between sentences, it is also required to examine word-level connections in the whole text that may have meaningful impacts in determining the congruity of the text. The sentence embedding for the whole text is fed into hidden layers of the neural network. Finally, three sequential layers of neural network conclude the model, combining the output of all previous paths of hidden layers to predict the final output. The final layers aim to determine the congruence of sentences and detect the transformation of reader's viewpoint after reading the punchline.

The detailed architecture of the model is examined below:

- **For sentence input**
- Dense Layer with 40 Neurons and leakyReLU activation function
- (Concatenate split sentences)

- **For whole text input**
- Dense Layer with 60 Neurons and relu activation function

- (Concatenate whole text and full sentences)

- **For Combined input**
- Dense Layer with 200 Neurons and LeakyReLU
- Dense Layer with 180 Neurons and relu
- Dense Layer with 160 Neurons and LeakyReLU
- Dense Layer with 140 Neurons and relu
- Dense Layer with 120 Neurons and LeakyReLU
- Dense Layer with 100 Neurons and relu
- Dense Layer with 80 Neurons and LeakyReLU
- Dense Layer with 60 Neurons and relu
- Dense Layer with 40 Neurons and LeakyReLU
- Dense Layer with 20 Neurons and relu
- Output Layer with 1 Neuron and sigmoid

The architecture of the model suggests a network capable of extracting relevant global as well as contextual information; the decreasing nature of the final hidden layers allows for each step to isolate relevant information rather than evaluate all features, increasing the significance of the final result before applying a sigmoid activation function for classification. As rectified linear units can encounter the dying relu problem, LeakyReLUs have been added to combat this issue.

Evaluation & Results

After training the model on 160,000 and test it on 40,000 sentences of the novel dataset proposed in this paper. S-COLBERT has achieved quite impressive results comparing to the amount of complexity reduction in the overall architecture. As we can see in the figure below, S-Colbert achieved precision of **91.69%** making it the second best model for the problem proposed after the large COLBERT model. This is a quite satisfying result as it is dramatically smaller than COLBERT. However, following Accuracy, Recall, and most importantly F1; as it is the main metric for this task(cite this), S-COLBERT has achieved **91.55%**, **91.42%**, and **91.56%** respectively, placing it at the third place in comparison to the other model.

Method	Configuration	Accuracy	Precision	Recall	F1
Decision Tree	N/A	0.786	0.769	0.821	0.794
SVM	sigmoid, gamma=1.0	0.872	0.869	0.88	0.874
Multinomial NB	alpha=0.2	0.876	0.863	0.902	0.882
XGBoost	N/A	0.72	0.753	0.777	0.813
XLNet	XLNet-Large-Cased	0.916	0.872	0.973	0.92
COLBERT	N/A	0.982	0.99	0.974	0.982
Proposed		**0.9155**	**0.9142**	**0.9169**	**0.9156**

Table Reference [9]

Further, it is essential to mention that because of the complexity reduction and the modification in the architecture we are proposing; the model's computational power required was significantly lower than other accurate models in the figure above.

That being said, the embeddings computation took only **1773.8s** using two GPU T4 with 15 gigabyte each, which is very low compared to the results. The main issue we have faced throughout this stage was the high need for memory as the bert model is memory hungry. On the other hand, model training took 3169.1s using only a CPU of 4-cores with 30 gigabytes of memory. This was essential as the embeddings were still large considering the big dataset we are training on. Hence, the overall model training took approximately 82.3 minutes which is very efficient. Moreover, the final trained model was light and had a size of less than 5 megabytes.

Future Work

Various improvements can be made to increase the overall performance of the model, starting with the legitimacy of the data. Due to the large nature of the dataset as well as the recency of its publication one must question the validity of the labelling. It is not possible to ensure each of the 200,000 short texts is labelled correctly and this could lead to issues when training and evaluating the model. A dataset which has been verified to be entirely legitimate and correct is needed to guarantee no issues within the output stem from incorrect data practices. Additionally, more pre-processing of the data can be done to improve performance. Standard NLP practices such as stop-words, stemming, lemmatization or text normalisation. While it is not guaranteed that performing these operations will bring forth better output, they need to be evaluated nonetheless.

One must also consider the linguistic structure of other theories on Humor, While the current approach has shown to outperform other models significantly it is entirely possible that a different theory of the structure of humour could lead to even better results. Significant amounts of research can be done into what theories are most prevalent and which seem to make the most sense in terms of applying it to the model architecture. The Incongruous juxtaposition theory for example could be considered as the moment of realisation of incongruity between a concept involved in a certain situation and the reality within relation to that concept [8]. One could interpret this as identifying the moment in which the general distribution of a sentence suddenly changes drastically, a concept a model could certainly learn.

Currently the scope of the sentences was limited by an upper bound of words for simplicity purposes. Assuming no issues with computational power one could explore the effects of using much longer sentences, perhaps even stories. In this case it is likely that the text would be split into more than 2 sentences as capturing local context relies primarily on identifying relevant smaller regions. Including longer sentences within the data could also lead the model to be more robust toward non-humorous text, as it learns and familiarises itself with more information on the global structure of a joke.

It would also be interesting to explore the application of humour detection models in different languages than English. As other languages have different syntax and meaning entirely, the architecture may need to be modified significantly. To gain an insight into the way humour is developed in linguistics through different languages can help identify which parts of sentences across any language tend to be most relevant for humour identification. There are also applications of this in translating humour, a joke written in English may well have to be set up entirely differently in Chinese; extracting the relevant structure in each language can help identify how to convert a joke from one language to another without losing either the original meaning or the humour contained in it.

Altering the model architecture can also be done to attempt to improve model performance. Additional layers such as LSTM layers can be used to retain certain patterns within the language and structure of sentences. Batch normalisation layers could be added to stabilise the network and allow for the learning rate to be increased. This can help with model training and also permits the use of larger models. On the other hand, it has been shown that implementing dropout layers throughout the network can achieve similar results [1]. Additionally, when examining the current state of the art literature it seems that global average pooling 1D layers are typically used to reduce the dimensionality of the feature space before feeding them forward to the dense layers. Overall these approaches can be explored relatively easily given enough computational power.

It must be mentioned that the results used to evaluate and compare different tree based models in COLBERT PAPER were derived from evaluations using tf-idf vector representations rather than the novel BERT approach [1]. Due to this, the comparative performance of the tree models may not be entirely reflective when compared to approaches utilising the BERT embeddings. It is important to train and test the various tree models on the correct embeddings to ensure unbiased and legitimate results and comparison.

Conclusion

Overall, on 40,000 tests, S-COLBERT achieved 91.55%, 91.42% and 91.56%, in terms of accuracy, recall and F1, ranking it at third place in comparison to other models. In terms of precision it ranked second with a score of 91.69%. Additionally to these results the model is a lot simpler and smaller than COLBERT, generating embeddings in 1773.8s, using GPU T4 with 15GB of memory each. Model training took 3169.1s using a CPU of 4 cores with 30GB of memory. The model size was also very light at less than 5MB. The results are favourable in spite of the reduction of the model's complexity, suggesting the solution could be vastly improved with access to greater computational resources.

Bibliography

[1] Annamoradnejad, I. & Zoghi, G., (27 April, 2020). ColBERT: Using BERT Sentence Embedding in Parallel Neural Networks for Computational Humor. *Papers With Code*. Accessed at https://arxiv.org/pdf/2004.12765v7.pdf

[2] Chen, T. & Guestrin, C., (9 March, 2016). XGBoost: A Scalable Tree Boosting System. *Papers With Code*. Accessed at https://arxiv.org/pdf/1603.02754v3.pdf

[3] Papers With Code, (2023). Humor Detection on 200k Short Texts for Humor Detection. *Papers With Code*. Accessed at https://paperswithcode.com/sota/humor-detection-on-200k-short-texts-for-humor-1

[4] Sandling, J., (n.d.). Script-Based Semantic Theory of Humour. *Jonathan Sandling*. Accessed at https://jonathansandling.com/script-based-semantic-theory-of-humour/?utm_content=cmp-true

[5] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. & Le, Q. V., (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. *Papers With Code.* Accessed at https://arxiv.org/pdf/1906.08237v2.pdf

[6] Hugging Face. (n.d.). Gaunernst/bert-small-uncased. Hugging Face Model Hub. Retrieved April 26, 2023, from https://huggingface.co/gaunernst/bert-small-uncased/tree/main

[7] Deepcontractor. (2021). 200K Short Texts for Humor Detection [Data set]. Kaggle. http://www.kaggle.com/datasets/deepcontractor/200k-short-texts-for-humor-detection

[8] Knox, T. (2021, March 2). Explaining the Incongruity Theory of Comedy. Owlcation. https://owlcation.com/humanities/Explaining-the-Incongruity-theory-of-Comedy.

[9] Papers with Code. (n.d.). Humor Detection - Papers With Code. Retrieved April 26, 2023, from https://paperswithcode.com/task/humor-detection/codeless